

PATENT
EMCR:070
EMC-00-174

APPLICATION FOR UNITED STATES LETTERS PATENT
for

ADAPTIVE BIT RATE CONTROL FOR RATE REDUCTION OF MPEG CODED
VIDEO

by

Seyfullah H. Oguz
Sorin Faibish
Wayne W. Duso

EXPRESS MAIL MAILING LABEL NUMBER EL291391701 US

0088212 8YP0011.DOC

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to processing and storage of compressed visual data, and in particular the processing and storage of compressed visual data for bit rate reduction.

2. Background Art

It has become common practice to compress audio/visual data in order to reduce the capacity and bandwidth requirements for storage and transmission. One of the most popular audio/video compression techniques is MPEG. MPEG is an acronym for the Moving Picture Experts Group, which was set up by the International Standards Organization (ISO) to work on compression. MPEG provides a number of different variations (MPEG-1, MPEG-2, etc.) to suit different bandwidth and quality constraints. MPEG-2, for example, is especially suited to the storage and transmission of broadcast quality television programs.

For the video data, MPEG provides a high degree of compression (up to 200:1) by encoding 8 x 8 blocks of pixels into a set of discrete cosine transform (DCT) coefficients, quantizing and encoding the coefficients, and using motion compensation techniques to encode most video frames as predictions from or between other frames. In particular, the encoded MPEG video stream is comprised of a series of groups of pictures (GOPs), and each GOP begins with an independently encoded (intra) I frame and may include one or more following P frames and B frames. Each I frame can be decoded without information from any preceding and/or following frame. Decoding of a P frame requires information from a preceding frame in the GOP. Decoding of a B frame requires information from both a preceding and a following frame in the GOP. To minimize

1 Analysis (Including DVB and ATSC)," Tektronix Inc., 1997, incorporated herein by
2 reference.

3 MPEG-2 provides several optional techniques that allow video coding to be
4 performed in such a way that the coded MPEG-2 stream can be decoded at more than one
5 quality simultaneously. In this context, the word "quality" refers collectively to features
6 of a video signal such as spatial resolution, frame rate, and signal-to-noise ratio (SNR)
7 with respect to the original uncompressed video signal. These optional techniques are
8 known as MPEG-2 scalability techniques. In the absence of the optional coding for such
9 a scalability technique, the coded MPEG-2 stream is said to be nonscalable. The MPEG-
10 2 scalability techniques are varieties of layered or hierarchical coding techniques, because
11 the scalable coded MPEG-2 stream includes a base layer that can be decoded to provide
12 low quality video, and one or more enhancement layers that can be decoded to provide
13 additional information that can be used to enhance the quality of the video information
14 decoded from the base layer. Such a layered coding approach is an improvement over a
15 simulcast approach in which a coded bit stream for a low quality video is transmitted
16 simultaneously with an independently coded bit stream for high quality video. The use of
17 video information decoded from the base layer for reconstructing the high quality video
18 permits the scalable coded MPEG-2 stream to have a reduced bit rate and data storage
19 requirement than a comparable simulcast data stream.

20 The MPEG-2 scalability techniques are useful for addressing a variety of
21 applications, some of which do not need the high quality video that can be decoded from
22 a nonscalable coded MPEG stream. For example, applications such as video
23 conferencing, video database browsing, and windowed video on computer workstations

003627 9905260

1 8x8 block. For low bit rates, however, the stuffing represents a waste of information
2 capacity resulting in a significant loss of picture quality. Efforts toward improving the bit
3 rate control has focused on more sophisticated methods of estimating the complexity of
4 image segments in comparison to average complexity in order to more precisely allocate
5 bits to the image segments to obtain a rather constant visual picture quality and thus
6 minimal degradation in picture quality. One solution was to perform two-pass encoding,
7 which cannot be performed in real time. In a first pass, the video sequence is encoded
8 with constant-bit-rate (CBR) encoding, while statistics concerning coding complexity are
9 gathered. Next, the first pass data is processed to prepare control parameters for the
10 second pass, which performs the actual VBR compression. See Westerink et al., "Two-
11 pass MPEG-2 variable-bit-rate encoding," IBM J. Res. Develop., Vol. 43, No. 4, July
12 1999, pp. 471-488. Real-time, single-pass encoding has also been proposed which is said
13 to adapt to the complexity of the image segments. However, substantial processing
14 resources are required for good performance from a single-pass encoder. See Mohsenian
15 et al., "Single-pass constant- and variable-bit-rate MPEG-2 video compression," IBM J.
16 Res. Develop., Vol. 43, No. 4, July 1999, pp. 489-509.

17 SUMMARY OF THE INVENTION

18
19 The basic objective of the present invention is to provide an improved bit rate
20 control for producing reduced-quality MPEG coded video from original quality MPEG
21 coded video. It is desired to produce the reduced-quality MPEG coded video at a given
22 constant or variable bit rate with a minimum of computing resources, yet ensure that the
23 reduced-quality MPEG coded video will have a relatively constant visual picture quality
24 over the picture segments and thus minimal degradation in picture quality.

1 In accordance with one aspect, the invention provides a method of producing
2 reduced-quality MPEG coded video from original-quality MPEG coded video. The
3 original-quality MPEG coded video includes a set of non-zero AC discrete cosine
4 transform (DCT) coefficients for 8x8 blocks in frames of the original-quality MPEG
5 coded video. The reduced-quality MPEG coded video also has frames of 8x8 blocks.
6 Each frame in the reduced-quality MPEG coded video has a corresponding frame in the
7 original-quality MPEG coded video, and each 8x8 block in each frame of the reduced-
8 quality MPEG coded video has a corresponding block in a corresponding frame in the
9 original-quality MPEG coded video. The method includes the steps of selecting non-zero
10 AC DCT coefficients from corresponding blocks in the original-quality MPEG coded
11 video to be included in the blocks of the reduced-quality MPEG coded video so that each
12 8x8 block in each frame of the reduced-quality MPEG coded video has a number of bits
13 encoding non-zero AC DCT coefficients that is generally proportional to the number of
14 bits encoding non-zero AC DCT coefficients for the corresponding 8x8 block in the
15 corresponding frame of original-quality MPEG coded video.

16 In accordance with another aspect, the invention provides a method of producing
17 reduced-quality MPEG coded video from original-quality MPEG coded video. The
18 original-quality MPEG coded video includes a set of non-zero AC discrete cosine
19 transform (DCT) coefficients for 8x8 blocks in frames of the original-quality MPEG
20 coded video. The reduced-quality MPEG coded video also has frames of 8x8 blocks.
21 Each frame in the reduced-quality MPEG coded video has a corresponding frame in the
22 original-quality MPEG coded video, and each 8x8 block in each frame of the reduced-
23 quality MPEG coded video has a corresponding block in a corresponding frame in the

1 quality MPEG-2 coded video. The steps for each frame include (a) computing a moving
2 average of the size of the corresponding frame in the original-quality MPEG-2 coded
3 video; (b) computing a scale factor from the moving average of the size of the
4 corresponding frame in the original-quality MPEG-2 coded video and a desired size of
5 said each frame of the reduced-quality MPEG-2 coded video; and (c) for each 8x8 block
6 in said each frame: (i) determining the number of bits used in encoding non-zero AC
7 DCT coefficients in the corresponding block of original-quality MPEG-2 coded video;
8 (ii) computing a number of bits available for encoding AC DCT coefficients in the
9 original-quality MPEG-2 coded video by scaling the number of bits used in encoding
10 non-zero AC DCT coefficients in the corresponding block of original-quality MPEG-2
11 coded video with a scale factor, and (iii) selecting non-zero AC DCT coefficients in a
12 parsing order from the corresponding block in the original-quality MPEG-2 coded video
13 to be included in said each block of the reduced-quality MPEG-2 coded video until the
14 number of bits available for encoding the AC DCT coefficients in the block in the
15 reduced-quality encoded video is not sufficient for encoding, in the block of the reduced-
16 quality MPEG-2 coded video, any more of the AC DCT coefficients in the corresponding
17 block of original-quality MPEG-2 coded video.

19 BRIEF DESCRIPTION OF THE DRAWINGS

20 Other objects and advantages of the invention will become apparent upon reading
21 the following detailed description with reference to the accompanying drawings, in
22 which:

23 FIG. 1 is a block diagram of a data network including a video file server
24 implementing various aspects of the present invention;

1 FIG. 2 is a flowchart of a procedure executed by a stream server computer in the
2 video file server of FIG. 1 to service client requests;

3 FIG. 3 is a flowchart of a procedure for splicing MPEG clips;

4 FIG. 4 is a flowchart of a procedure for seamless video splicing of MPEG clips;

5 FIG. 5 is a more detailed flowchart of the procedure for seamless video splicing
6 of MPEG clips;

7 FIG. 6 is a continuation of the flowchart begun in FIG. 5;

8 FIG. 7 is a timing diagram showing a timing relationship between video
9 presentation units (VPUs) and associated audio presentation units (APUs) in an original
10 MPEG-2 coded data stream;

11 FIG. 8 is a timing diagram showing a timing relationship between video
12 presentation units (VPUs) and associated audio presentation units (APUs) for a fast-
13 forward trick-mode stream;

14 FIG. 9 is a flowchart of a procedure for selection and alignment of audio
15 presentation units (APUs) in the fast-forward trick-mode stream;

16 FIG. 10 is a flowchart of a procedure for producing a trick-mode MPEG-2
17 transport stream from a regular MPEG-2 transport stream (TS);

18 FIG. 11 is a diagram illustrating relationships between the MPEG discrete cosine
19 transform (DCT) coefficients, spatial frequency, and the typical zig-zag scan order;

20 FIG. 12 is a diagram illustrating a relationship between an MPEG-2 coded bit
21 stream and a reduced-quality MPEG-2 coded bit stream resulting from truncation of high-
22 order DCT coefficients;

1 FIG. 13 is a flowchart of a procedure for scaling MPEG-2 coded video using a
2 variety of techniques;

3 FIG. 14 is a flowchart of a procedure for signal-to-noise ratio scaling MPEG-2
4 coded video using a frequency-domain low-pass truncation (FDSNR_LP) technique;

5 FIG. 15 is a flowchart of a procedure for signal-to-noise ratio scaling MPEG-2
6 coded video using a frequency-domain largest-magnitude coefficient selection
7 (FDSNR_LM) technique;

8 FIG. 16 is a flowchart of a procedure that selects one of a number of techniques
9 for finding a certain number "k" of largest values out of a set of "n" values;

10 FIG. 17 is a flowchart of a procedure for finding a certain number "k" of largest
11 values from a set of "n" values, which is used in the procedure of FIG. 16 for the case of
12 $k \ll \frac{1}{2} n$;

13 FIG. 18 is a diagram of a hash table and associated hash lists;

14 FIG. 19 is a flowchart of a procedure for finding a certain number "k" of values
15 that are not less than the smallest of the "k" largest values in a set of "n" values beyond a
16 certain amount.

17 FIG. 20 is a flowchart of modification of the procedure of FIG. 15 in order to
18 possibly eliminate escape sequences in the (run, level) coding of the largest magnitude
19 coefficients;

20 FIG. 21 is a flowchart of a subroutine called in the flowchart of FIG. 20 in order
21 to possibly eliminate an escape sequence;

22 FIG. 22 is a first portion of a flowchart of a procedure for scaling an MPEG-2
23 coded video data stream using the modified procedure of FIG. 20 while adjusting the

1 parameter "k" to achieve a desired bit rate, and adjusting a quantization scaling factor
2 (QSF) to achieve a desired frequency of occurrence of escape sequences;

3 FIG. 23 is a second portion of the flowchart begun in FIG. 22;

4 FIG. 24 is a simplified block diagram of a volume containing a main file, a
5 corresponding fast forward file for trick mode operation, and a corresponding fast reverse
6 file for trick mode operation;

7 FIG. 25 is a more detailed block diagram of the volume introduced in FIG. 24;

8 FIG. 26A is a diagram showing video file access during a sequence of video
9 operations including transitions between the main file, the related fast forward file, and
10 the related fast reverse file;

11 FIG. 26B shows a script of a video command sequence producing the sequence of
12 video play shown in FIG. 26A;

13 FIG. 27 is a table of read and write access operations upon the volume of FIG. 24
14 and access modes that are used for the read and write access operations;

15 FIG. 28 is a hierarchy of video service classes associated with the fast forward file
16 and the fast reverse file in the volume of FIG. 25;

17 FIG. 29 shows a system for modifying and combining an MPEG-2 audio-visual
18 transport stream with an MPEG-2 closed-captioning transport stream to produce a
19 multiplexed MPEG-2 transport stream having the same bit rate as the original MPEG-2
20 audio-visual transport stream;

21 FIG. 30 is a diagram showing blocks in a frame of original-quality MPEG coded
22 video being proportionally reduced to obtain a frame of reduced-quality MPEG coded
23 video in an adaptive bit rate reduction method;

FIG. 31 shows a graph of frame size in bits per frame as a function of frame number for original-quality MPEG coded video and corresponding reduced-quality MPEG coded video obtained using the adaptive bit rate reduction method introduced in FIG. 30;

FIG. 32 shows a graph of frame size in bits per frame as a function of frame number for original-sized MPEG I frames and corresponding reduced-size I frames for trick file generation using the adaptive bit rate reduction method introduced in FIG. 30;

FIG. 33 is a first sheet of a flowchart for programming a digital computer to implement the adaptive bit rate reduction method introduced in FIG. 30;

FIG. 34 is a second sheet of the flowchart begun in FIG. 33;

FIG. 35 is a third sheet of the flowchart begun in FIG. 33;

FIG. 36 is a subroutine for programming a digital computer to determine a coefficient bit rate reduction factor for a reduction from an MPEG source having a known constant bit rate;

FIG. 37 is a subroutine for programming a digital computer to determine a coefficient bit rate reduction factor for a reduction from an MPEG source having an unknown or variable bit rate; and

FIG. 38 is a block diagram illustrating the use of the adaptive bit rate reduction method introduced in FIG. 30 for the real-time production of a reduced-quality MPEG-2 stream from an original-quality MPEG-2 source.

While the invention is susceptible to various modifications and alternative forms, specific embodiments thereof have been shown by way of example in the drawings and will be described in detail. It should be understood, however, that it is not intended to

1 limit the form of the invention to the particular forms shown, but on the contrary, the
2 intention is to cover all modifications, equivalents, and alternatives falling within the
3 scope of the invention as defined by the appended claims.

5 DESCRIPTION OF ILLUSTRATIVE EMBODIMENTS

6 With reference to FIG. 1, there is shown a block diagram of a data network 20
7 linking a number of clients 21, 22, 23 to a video file server 24 implementing various
8 aspects of the present invention. The video file server 24 includes at least one stream
9 server computer 25 and a data storage system 26. The stream server computer 25 has a
10 processor 27 and a network link adapter 28 interfacing the processor to the data network
11 20. The processor 27 executes a data streaming program 29 in memory 30 in order to
12 stream MPEG coded video in real-time to the clients.

13 Client requests for real-time video are placed in client play lists 31 in order to
14 schedule in advance video file server resources for the real-time streaming of the MPEG
15 coded video. The play lists 31 specify a sequence of video clips, which are segments of
16 MPEG-2 files 32, 33 in data storage 34 of the data storage system 26. The stream server
17 processor 27 accesses a client play list in advance of the time to begin streaming MPEG
18 coded video from a clip, and sends a video prefetch command to a storage controller 35
19 in the data storage system 26. The storage controller responds to the video prefetch
20 command by accessing the clip in the data storage 34 to transfer a segment of the clip to
21 cache memory 36. When the video data of the segment needs to be sent to the client, the
22 stream server processor 27 requests the data from the storage controller 35, and the
23 storage controller immediately provides the video data from the cache memory 36.
24 Further details regarding a preferred construction and programming of the video file

1 invention permits a PC workstation 23 to perform the decoding and display in real-time
2 by execution of a software program. An operator can view the video content in a display
3 window 39 in a fast-forward or fast-reverse mode, stop at and resume from freeze frames
4 that are valid "in points" and "out points" for seamless splicing, and select an in-point
5 and out-point for a next segment to be included in the play list. The stream server
6 computer 25 could also include a seamless splicing program 40 providing seamless
7 transitions between video segments that are contiguous in a play list and are from
8 different video clips.

9 For seamless splicing, it is often necessary to reduce the bitrate for one or more
10 frames at the end of a first segment prior to splicing to a second segment. In this case the
11 bitrate must be reduced to avoid buffer overflow as a result of displaying the original
12 frames at the end of the first segment. One method of reducing the bitrate is to insert a
13 freeze frame at the end of the first segment, but this has the disadvantage of introducing
14 distortion in the temporal presentation of the frames and precluding frame accuracy. A
15 less disruptive method is to use the present invention for reducing the bitrate for a lower-
16 quality presentation of one or more frames at the end of the first segment.

17 The present invention can also reduce the bit transmission rate and storage
18 requirements for MPEG-2 applications by altering the video quality. For example,
19 different clients may present different bandwidth access requests for video from
20 nonscalable MPEG-2 files 32, 33 in the video file server. Also, temporary network
21 congestion may limit the bandwidth available to satisfy a request for real-time streaming
22 of video data. In each case, the present invention can alter the video quality to meet the
23 desired or available bandwidth to satisfy the request.

1 With reference to FIG. 2, there is shown a flowchart of a procedure executed by a
2 stream server computer in the video file server of FIG. 1 to service client requests. In a
3 first step 50, execution branches to step 51 when a client request is not a request for real-
4 time streaming. If the request is a request to input a new MPEG-2 file, then execution
5 branches to step 52 to input the new MPEG-2 file and to create a reduced-quality version
6 of the MPEG-2 file as available resources permit. If the request is not a request to input a
7 new MPEG-2 file, then execution continues from step 51 to step 53. In step 53,
8 execution branches to step 54 if the request is for play list editing. In step 54, the client
9 may browse through the reduced-quality MPEG file to select in-points and out-points of
10 clips to be spliced.

11 In step 50, when the request is for real-time streaming, then execution branches to
12 step 55. In step 55, if there is network congestion so that there is insufficient bandwidth
13 to transmit a stream of original-quality MPEG-2 coded video, then execution branches to
14 step 56 to stream compressed video from the reduced-quality MPEG file. If no reduced-
15 quality MPEG file is available for the desired clip, then the reduced-quality MPEG coded
16 video to be streamed is produced in real-time from the original-quality MPEG-2 coded
17 video. There are also applications, such as the display of spatially down-sampled video
18 in a small display window (39 in FIG. 1), for which the client may request reduced-
19 quality MPEG coded video. In this case, in the absence of network congestion, execution
20 will continue from step 55 to step 57, and branch from step 57 to step 56 for streaming of
21 reduced-quality MPEG coded video to the client.

22 Reduced-quality MPEG coded video is also useful for "trick-mode" operation.
23 Trick-mode refers to fast forward or fast reverse display of video, in a fashion analogous

1 to the fast forward and fast reverse playback functions of a video cassette recorder
2 (VCR). The problem with trick-mode operation is that the speed of the MPEG stream
3 cannot simply be speeded up because the transmission bandwidth would be excessive and
4 a conventional MPEG-2 decoder will not be able to handle the increased data rate or even
5 if the decoder would have been able to support the increased data rate, such a change in
6 the original operating conditions is not allowable. For this reason, in trick-mode, neither
7 the original display rate of 29.97 frames per second (for NTSC or 25 frames per second
8 for PAL) nor the original transport stream (TS) multiplex rate should change. Nor is it
9 possible to simply decimate frames since only the I frames are independently coded, and
10 the P frames and B frames need the content of certain other frames for proper decoding.
11 The I frames typically occur once for every 15 frames. Assuming that this convention is
12 followed in the encoding process, it would be possible to preserve and play each I frame
13 from each and every group of pictures (GOP), resulting in a 15 times slower temporal
14 sampling rate, or a 1 to 15 speeding up of motion if the I frames only are played back at
15 the nominal NTSC rate of approximately 30 frames per second. Consequently, the
16 content of a 60 minutes duration clip will be covered in 4 minutes. Unfortunately the
17 average information content per frame for the I frames is more than four times the
18 average information content of the P and B frames. Therefore, the trick-mode cannot be
19 implemented simply by transmitting only the I frames for a speed-up by a factor of 15,
20 because this would need an increase in the TS multiplex rate over the nominal rate.

21 In particular, the average information content of an I frame has been measured to
22 be about 56,374.6 bytes. If the I frames only are transmitted at the standard NTSC rate,
23 then the bit transmission rate would be: $8(\text{bits per byte}) * 56,374.6(\text{bytes per frame}) *$

1 29.97(frames per sec.) or about 13,516,374.1 bits per second only for the video stream,
2 which is significantly above - almost 3.38 times - the original rate of 4 megabits per
3 second used in this test. This calculation, being based on an average quantity, is ignoring
4 the indispensable need for an actually higher transport rate to provide some safety margin
5 to handle short-term-sustained large size I frame chains (bursts) which practically always
6 happen. Clearly, some form of modification in the trick-mode operation definition is
7 required to handle this problem and pull the bit-rate requirement down to the nominal 4
8 megabits per second.

9 Two degrees of freedom are available to achieve such a reduction in the required
10 bit-rate for trick-mode operation. The first is I frame compression quality and the second
11 is a motion speed-up ratio. With respect to compression quality, it is well known that
12 human observers' perception of image detail degrades with increasing motion speed of
13 objects in the scene. Based on this fact, the type of D pictures were introduced in MPEG-
14 1 video syntax for fast visible (forward or reverse) search purposes. (See ISO/IEC 11172-
15 2: 1993 Information Technology - Coding of moving pictures and associated audio for
16 digital storage media at up to about 1.5 Mbits/s - Part 2: Video, Annex D.6.6. Coding D-
17 Pictures, p. 102). D pictures make use of only the DC coefficients in intra coding to
18 produce very low quality (in terms of SNR) reproductions of desired frames which were
19 judged to be of adequate quality in fast search mode.

20 In order to provide support for enhanced quality trick-mode operation, the quality
21 of the original I frames can be reduced by the preservation of just a sufficient number of
22 AC DCT coefficients to meet the bit-rate limitation. Based on experiments with two
23 standard video test sequences (one encoded at 15 Mbits/sec. and the other at 24

1 Mbits/sec. and both with I frames only), it is observed that the bandwidth for I frames can
2 be scaled to one half by keeping about 9 lowest order AC coefficients and eliminating the
3 rest. This scheme provides good quality even at the full spatial and temporal resolution,
4 much better than D pictures.

5 The inherent speed-up ratio lower bound imposed by the GOP structure can be
6 relaxed and further lowered by freeze (P) frame substitution in between genuine (SNR
7 scaled or non-scaled) I frames. The maximum number of freeze frames that can be
8 inserted before visually disturbing motion jerkiness occurs, is very likely to depend
9 heavily on the original GOP structure (equivalently the separation between I frames of
10 the original sequence) and the original amount of motion in the clip. However, 1, 2 or 3
11 freeze frame substitutions in between genuine I frames present reasonable choices which
12 will yield speed-up ratios of 1 to 7.5, 1 to 5 and 1 to 3.75 respectively instead of the 1 to
13 15 speed-up ratio provided by the genuine I frames only implementation. (These ratios
14 are computed by a first-order approximation that neglects a slight increase in bandwidth
15 required by the consecutive freeze frames, which are inserted in between genuine I
16 frames and can typically be made very small in size in comparison to the average size of
17 a genuine I frame. Therefore, the insertion of 1, 2, 3 freeze frames will result in
18 bandwidth reductions of 2 to 1, 3 to 1 and 4 to 1 respectively. The accuracy of this
19 approximation degrades as more consecutive freeze frames and/or SNR scaling is
20 employed.) An easy way to see the validity of these approximate figures is to note for
21 example that in the case of 1 freeze frame insertion, the total presentation time of the
22 trick-mode clip for an originally 60 minutes duration asset will increase from 4 minutes
23 to 8 minutes. Since due to the underlying assumption of the first-order approximation

1 number of freeze frames between the reduced-quality I frames. If a trick-mode operation
2 is not requested in step 58, then execution continues from step 58 to step 63. In step 63,
3 the stream server computer streams original-quality MPEG-2 coded data to the client.
4 Further details regarding trick-mode operation are described below with reference to
5 FIGs. 7 to 10.

6 FIGs. 3 to 6 show further details regarding use of the present invention for MPEG
7 splicing. In particular, reduced-quality frames are substituted for the freeze frames used
8 in the seamless splicing procedure found in the common disclosure of Peter Bixby et al.,
9 U.S. application Ser. 09/539,747 filed March 31, 2000; Daniel Gardere et al., U.S.
10 application Ser. 09/540,347 filed March 31, 2000; and John Forecast et al., U.S.
11 application Ser. 09/540,306 filed March 31, 2000; which are all incorporated by reference
12 herein. The common disclosure in these U.S. applications considered pertinent to the
13 present invention is included in the written description below with reference to FIGs. 3 to
14 6 in the present application (which correspond to FIGs. 19, 22, 23, and 24 in each of the
15 cited U.S. applications).

16 FIG. 3 shows a basic procedure for MPEG splicing. In the first step 121, the
17 splicing procedure receives an indication of a desired end frame of the first clip and a
18 desired start frame of the second clip. Next, in step 122, the splicing procedure finds the
19 closest I frame preceding the desired start frame to be the In Point for splicing. In step
20 123, the splicing procedure adjusts content of the first clip near the end frame of the first
21 clip and adjusts content of the second clip near the In Point in order to reduce
22 presentation discontinuity (due to decoder buffer underflow) and also to prevent decoder
23 buffer overflow when decoding the spliced MPEG stream. Finally, in step 124, the

1 be a problem of video buffer management if a duration of time $DTS_{L1}-T_e$ is different from
2 a duration of time $DTS_{F2}-PCR_{e2}$ minus one video frame (presentation) interval, where
3 DTS_{L1} is the DTS at the end of the first clip and indicates the time at which the video
4 decoder buffer is emptied of video data from the first clip, T_e is the time at which the last
5 video frame's data is finished being loaded into the video decoder buffer, DTS_{F2} is the
6 DTS of the first frame of the second clip, and PCR_{e2} is the PCR of the second clip
7 extrapolated from the value of the most recent received genuine PCR record, to the first
8 byte of the picture header sync word of the first video frame in the clip to start. The
9 extrapolation adjusts this most recently received genuine PCR record value by the
10 quotient of the displacement in data bits of the clip from the position where it appears in
11 the second clip to the position at which video data of the first frame of the second clip
12 begins, divided by the data transmission bit rate for transmission of the clip to the
13 decoder. Because the time PCR_{e2} must immediately follow T_e , there will be a gap in the
14 decoding and presentation of video frames if $DTS_{F2}-PCR_{e2}$ is substantially greater than
15 $DTS_{L1}-T_e$ plus one video frame interval. In this case, the buffer will not be properly full
16 to begin decoding of the second clip one video frame interval after the last frame of the
17 first clip has been decoded. Consequently, either the second clip will be prematurely
18 started to be decoded or the decoder will be forced to repeat a frame one or more times
19 after the end of the display of the last frame from the first clip to provide the required
20 delay for the second clip's buffer build-up. In the case of a premature start for decoding
21 the second clip, a video buffer underflow risk is generated. On the other hand, in case of
22 repeated frames, the desired frame accuracy for scheduled play-lists is lost besides the
23 fact that neither a precise timing adjustment can be achieved through this procedure.

1 If $DTS_{F2}-PCR_{e2}$ is substantially less than $DTS_{L1}-T_e$ plus one video frame interval,
2 then the decoder will not be able to decode the first frame of the second clip at the
3 specified time DTS_{F2} because the last frame of the first clip will not yet have been
4 removed from the video buffer. In this case a video buffer overflow risk is generated.
5 Video buffer overflow may present a problem not only at the beginning of the second
6 clip, but also at a subsequent location of the second clip. If the second clip is encoded by
7 an MPEG-2 compliant encoder, then video buffer underflow or buffer overflow will not
8 occur at any time during the decoding of the clip. However, this guarantee is no longer
9 valid if the $DTS_{F2}-PCR_{e2}$ relationship at the beginning of the second clip is altered.
10 Consequently, to avoid buffer problems, the buffer occupancy at the end of the first clip
11 must be modified in some fashion. This problem is inevitable when splicing between
12 clips having significantly different ending and starting buffer levels. This is why the
13 Society of Motion Picture and Television Engineers (SMPTE) has defined some splice
14 types corresponding to well-defined buffer levels. (See SMPTE Standard 312M, entitled
15 "Splice Points for MPEG-2 Transport Streams," SMPTE Journal, Nov. 1998.) In order to
16 seamlessly splice the first clip to the second clip, the content of the first clip (towards its
17 end) is modified so that PCR_{e2} can immediately follow T_e (by one byte transmission
18 time) and DTS_{F2} can just follow DTS_{L1} (by one video frame presentation interval).

19 FIG. 4 shows a flow chart of a seamless video splicing procedure that attains the
20 desired condition just described above. In a first step 141, the first DTS of the second
21 clip is anchored at one frame interval later than the last DTS of the first clip in order to
22 prevent a video decoding discontinuity. Then, in step 142, the procedure branches
23 depending on whether the PCR extrapolated to the beginning frame of the second clip

1 falls just after the ending time of the first clip. If so, then the splice will be seamless with
2 respect to the original video content. Otherwise, the procedure branches to step 143. In
3 step 143, the content of the first clip is adjusted so that the PCR extrapolated to the
4 beginning frame of the second clip falls just after the ending time of the first clip.
5 Therefore the desired conditions for seamless video splicing are achieved.

6 With reference to FIG. 5, there is shown a more detailed flow chart of a seamless
7 video splicing procedure. In a first step 151, the procedure inspects the content of the
8 first clip to determine the last DTS/PTS of the first clip. This last DTS/PTS of the first
9 clip is designated DTS_{L1} . Next, in step 152, the procedure inspects the content of the first
10 clip to determine the time of arrival (T_e) of the last byte of the first clip. In step 153, the
11 procedure adds one frame interval to DTS_{L1} to find the desired first DTS location for the
12 second clip. The sum, designated DTS_{F1} , is equal to $DTS_{L1} + 1/FR$, where FR is the video
13 frame rate. In step 154, while keeping the DTS-PCR_e relationship unaltered for the
14 second clip, the procedure finds the time instant, designated T_s , at which the first byte of
15 the second clip should arrive at the decoder buffer. This is done by calculating
16 $T_{START} = DTS_{F2} - PCR_{e2}$, and $T_s = DTS_{F1} - T_{START}$.

17 Continuing in FIG. 6, in step 155, execution branches depending on whether T_s is
18 equal to T_e plus 8 divided by the bit rate. If not, then the clips to be spliced need
19 modification before concatenation, and execution branches to step 156. In step 156,
20 execution branches depending on whether T_s is less than T_e plus 8 divided by the bit rate.
21 If not, then there is an undesired gap in between the clips to be spliced, and execution
22 branches to step 157. In step 157, null packets are inserted into the clips to be spliced to
23 compensate for the gap. The gap to be compensated has a number of bytes, designated

1 successive video presentation units for video frames of type I, B, B, P, B respectively.
 2 The trick-mode stream has successive video presentation units for video frames of types
 3 I, F, F, I, F where "F" denotes a freeze P (or possibly B) frame. Each I frame and
 4 immediately following F frames produce the same video presentation units as a
 5 respective I frame in the original data stream of FIG. 7, and in this example, one in every
 6 15 frames in the original data stream is an I frame. Each freeze frame is coded, for
 7 example, as a P frame repeating the previous I frame or the previous P-type freeze-frame
 8 (in display order). In each freeze frame, the frame is coded as a series of maximum-size
 9 slices of macroblocks, with an initial command in each slice indicating that the first
 10 macroblock is an exact copy of the corresponding macroblock in the previous frame
 11 (achieved by predictive encoding with a zero valued forward motion compensation vector
 12 and no encoded prediction error), and two consequent commands indicating that the
 13 following macroblocks in the slice until and including the last macroblock of the slice are
 14 all coded in the same way as the first macroblock.

15 For trick-mode operation, there is also a problem of how to select audio
 16 presentation units (APU) to accompany the video presentation units that are preserved in
 17 the trick-mode stream. Because the video presentation units (VPU) have a duration of
 18 $(1/29.97)$ sec. or about 33.37 msec. and the audio presentation units (APU) have a
 19 duration of 24 msec., there is neither a one-to-one correspondence nor alignment between
 20 VPUs and APUs. In a preferred implementation, the audio content of a trick-mode clip is
 21 constructed as follows. Given the total presentation duration $(1/29.97)$ sec. or about
 22 33.37 msec. for a single video frame, it is clear that always at least one and at most two
 23 24 msec. long audio presentation units (APU) will start being presented during the end-

1 to-end presentation interval of each video frame. This statement refers to the original clip
2 and does not consider any audio presentation unit whose presentation is possibly
3 continuing as the video frame under consideration is just put on the display. The first of
4 the above mentioned possibly two audio presentation units will be referred to as the
5 aligned audio presentation unit with respect to the video frame under consideration. For
6 example, in FIG. 8, the APU_j is the aligned audio presentation unit with respect to the
7 VPU_i . Now, when the I frames are extracted and possibly SNR scaled and possibly
8 further interleaved with a number of freeze P frames in between them to produce the
9 trick-mode video packetized elementary stream (PES), the associated trick-mode audio
10 stream is constructed as follows. For each I type video frame presentation interval (and
11 for that matter also for freeze P type video frames) in this trick-mode clip, the above
12 stated fact of at least one (and at most two) audio presentation unit being started, holds.
13 Then for each I frame presentation interval in the trick-mode clip, once any possibly
14 previously started and continuing audio presentation unit ends, insert its aligned audio
15 presentation unit (from the original clip) and continue inserting APUs from the original
16 clip subsequent to the aligned one until covering the rest of the I frame presentation
17 interval and also any possibly following freeze P frame presentation intervals until
18 crossing into and overlapping (or less likely aligning) with the next I frame's presentation
19 interval. In FIG. 8, for example, the audio presentation units APU_j , APU_{j+1} , APU_{j+2} , and
20 APU_{j+3} are inserted, until crossing into and overlapping with the next I frame VPU_{i+15} .
21 Following APU_{j+3} is inserted APU_k , which designates the APU aligned with VPU_{i+15} in
22 the original stream. Clearly, the final alignment of (the aligned and consequent) audio
23 presentation units with respect to their associated I frames will be slightly different in the

1 are formed encapsulating the I frames. In step 185, the I frames are SNR scaled, for the
2 high speed cases of the trick-mode. In step 186, P-type freeze frames are inserted into the
3 stream of SNR scaled I frames (in between the scaled I frames), and valid PES packets
4 are formed for the trick-mode VES encapsulating the P-type freeze frames and the SNR
5 scaled I frames. Concurrently, in step 187, appropriate audio access units (from the
6 originally input MPEG-2 TS asset) are selected and concatenated based on the structure
7 of the VES being formed for the trick-mode clip, as described above with reference to
8 FIG. 9, and valid PES packet encapsulation is formed around these audio access units.
9 Finally, in step 188, the trick-mode TS stream is generated by multiplexing the trick-
10 mode VES from step 186 into a system information (SI) and audio PES carrying TS
11 skeleton including the audio PES packets from step 187.

12 FIGS 11 to 19 include details of the preferred techniques for truncating AC DCT
13 coefficients for producing low-quality MPEG coded video from original-quality MPEG-2
14 coded video. Most of these techniques exploit the fact that in the typical (default) zig-zag
15 scan order, the basis functions for the high-order AC DCT coefficients have an increasing
16 frequency content. FIG 11, for example, shows a matrix of the DCT coefficients C_{ij} . The
17 row index (i) increases with increasing vertical spatial frequency in a corresponding 8x8
18 coefficient block, and the column index (j) increases with increasing horizontal spatial
19 frequency in the corresponding 8x8 coefficient block. The coefficient C_{11} has zero
20 frequency associated with it in both vertical and horizontal directions, and therefore it is
21 referred to as the DC coefficient of the block. The other coefficients have non-zero
22 spatial frequencies associated with their respective basis functions, and therefore they are
23 referred to as AC coefficients. Each coefficient has an associated basis function $f_{ij}(x,y)$

1 that is separable into x and y components such that $f_{ij}(x,y) = f_i(y)f_j(x)$. The x and y
2 component functions $f_i(y)$ and $f_j(x)$ are shown graphically in FIG. 11 as cosine functions
3 in order to illustrate their associated spatial frequencies. In practice, the component
4 functions are evaluated at discrete points for the 64 pixel positions in the 8x8 blocks, so
5 that each of the DCT basis functions is an 8x8 array of real numbers. In particular, the
6 component functions are:

7
8 $f_i(y) = \text{SQRT}((2 - \delta_{i-1})/8) (\cos((\pi/8)(y-1/2)(i-1)))$ for $y=1, 2, 3, \dots, 8$

9 $f_j(x) = \text{SQRT}((2 - \delta_{j-1})/8) (\cos((\pi/8)(x-1/2)(j-1)))$ for $x=1, 2, 3, \dots, 8$

10
11 The heavy black line through the matrix of coefficients in FIG. 11 denotes the default
12 zig-zag scan order typically used for MPEG-2 encoding. Listed in this order, the
13 coefficients are $C_{11}, C_{12}, C_{21}, C_{31}, C_{22}, C_{13}, C_{14}, C_{23}, C_{32}, C_{41}, \dots, C_{86}, C_{77}, C_{68}, C_{78}, C_{87},$
14 C_{88} . The first coefficient in this zig-zag scan order is the DC coefficient C_{11} providing
15 the lowest spatial frequency content in the 8x8 block of pixels, and the last coefficient in
16 this zig-zag scan order is the coefficient C_{88} providing the highest spatial frequency
17 content in the 8x8 block of pixels.

18 FIG. 12 is a diagram illustrating a relationship between an original MPEG-2
19 coded bit stream 200 and a reduced-quality MPEG-2 coded bit stream 210 resulting from
20 truncation of high-order DCT coefficients from the original MPEG-2 coded bit stream.
21 Shown in the original MPEG-2 coded bit stream 200 is a portion of a video PES packet
22 including DCT coefficients for an 8x8 pixel block. The DCT coefficients include a
23 differentially coded DC coefficient 201, and three (run, level) events 202, 203, 204

1 For example, if the low-quality video stream will be downsampled by a factor of two in
 2 both the vertical and the horizontal directions, then the procedure removes any and all
 3 DCT coefficients having a row index (i) greater than four and any and all DCT
 4 coefficients having a column index (j) greater than four. This requires the decoding of
 5 the (run, level) coded coefficients to the extent necessary to obtain an indication of the
 6 coefficient indices. If a sufficient number of the original AC DCT coefficients are
 7 removed for a desired bandwidth reduction, then the scaling procedure is finished.
 8 Otherwise, execution branches from step 223 to step 224. Execution also continues from
 9 step 221 to step 224 if spatial downsampling is not intended.

10 In step 224, execution branches to step 225 if low-pass scaling is desired. Low-
 11 pass scaling requires the least computational resources and may produce the best results
 12 if the scaled, low-quality MPEG coded video is spatially downsampled. In step 225, the
 13 procedure retains up to a certain number of lowest-order AC DCT coefficients for each
 14 block and removes any additional DCT coefficients for each block. This is a kind of
 15 frequency domain signal-to-noise ratio scaling (FDSNR) that will be designated
 16 FDSNR_LP. A specific example of the procedure for step 225 will be described below
 17 with reference to FIG. 14.

18 Execution continues from step 224 to step 226 if low-pass scaling is not desired.
 19 In step 226, execution branches to step 227 if largest magnitude based scaling is desired.
 20 Largest magnitude based scaling produces the least squared error or difference between
 21 the original-quality MPEG-2 coded video and the reduced-quality MPEG coded video for
 22 a given number of nonzero AC coefficients to preserve, but it requires more
 23 computational resources than the low-pass scaling of step 225. More computational

resources are needed because if there are more nonzero AC coefficients than the desired number of AC coefficients for a block, then the (run, level) events must be decoded fully to obtain the coefficient magnitudes, and additional resources are required to find the largest magnitude coefficients. In step 227, the procedure retains up to a certain number of largest magnitude AC DCT coefficients for each block, and removes any and all additional AC DCT coefficients for each block. This is a kind of frequency domain signal-to-noise ratio scaling (FDSNR) that will be designated FDSNR_LM. A specific example of the procedure for step 227 will be described below with reference to FIG. 15.

If in step 226 largest magnitude based scaling is not desired, then execution continues to step 228. In step 228, execution branches to step 229 to retain up to a certain number of AC DCT coefficients that differ in magnitude from up to that number of largest magnitude AC DCT coefficients by no more than a certain limit. This permits a kind of approximation to FDSNR_LM in which an approximate search is undertaken for the largest magnitude AC DCT coefficients if there are more nonzero AC DCT coefficients than the desired number of AC DCT coefficients in a block. The approximate search can be undertaken using a coefficient magnitude classification technique such as a hashing technique, and the low-pass scaling technique can be applied to the classification level that is incapable of discriminating between the desired number of largest magnitude AC DCT coefficients. A specific example is described below with reference to FIG. 19.

With reference to FIG. 14, there is shown a flowchart of a procedure for scaling MPEG-2 coded video using the low-pass frequency-domain signal-to-noise (FDSNR_LP) scaling technique. This procedure scans and selectively copies components of an input

1 value is less than "k"). In step 251, the counter state l is set equal to the new counter
2 value $l+r+1$. Then, in step 252, the VLC just parsed (which will be a VLC encoding a
3 (run, level) event) is copied from the stream of original-quality MPEG-2 coded data to
4 the stream of reduced-quality MPEG-2 coded data. After step 252, execution loops back
5 to step 243 to continue the scanning of the stream of original-quality MPEG-2 coded
6 data.

7 In step 250, if the new counter value $l+r+1$ is equal to "k", then execution
8 branches from step 250 to step 253, to copy the VLC just parsed (which will be a VLC
9 encoding a (run, level) event) from the stream of original-quality MPEG-2 coded data to
10 the stream of reduced-quality MPEG-2 coded data. Next, in step 254, the procedure
11 copies an EOB marker to the stream of reduced-quality MPEG-2 coded data. After step
12 254, execution continues to step 249, where the procedure parses the input stream of
13 original-quality MPEG-2 coded data until the end of the next EOB marker, and the
14 procedure is finished for the current block.

15 FIG. 15 is a flowchart of a procedure for scaling MPEG-2 coded video using the
16 largest magnitude based frequency-domain signal-to-noise ratio (FDSNR_LM) scaling
17 technique. This routine is successively called, and each call processes coefficient data in
18 the input stream for one 8x8 block of pixels. No more than a specified number "k" of
19 largest magnitude AC DCT coefficients are copied for the block, and a different number
20 "k" can be specified for each block.

21 In a first step 261 in FIG. 15, the procedure parses and copies the input stream of
22 original-quality MPEG-2 coded data to the output stream of lower-quality MPEG-2 data
23 up to and including the differential DC coefficient variable-length code (VLC). Then in

1 step 262 all (run, level) event VLCs are parsed and decoded until and including the EOB
2 marker of the current block. The decoding produces coefficient identifiers and
3 corresponding quantization indices representing the quantized coefficient values. In step
4 263, the quantization indices are transformed to quantized coefficient values. In step 264,
5 the (quantized) coefficients are sorted in descending order of their magnitudes. In step
6 265, the first “k” coefficients of the sorted list are preserved and the last 63-k AC DCT
7 coefficients in the sorted list are set to zero. In step 266, (run, level) event formation and
8 entropy coding (VLC encoding) are applied to the new set of coefficient values. Finally,
9 in step 267, the VLCs resulting from step 266 are copied to the output stream until and
10 including the EOB marker.

11 The sorting step 264 of the FDSNR_LM procedure can consume considerable
12 computational resources. It is important to notice that not a full sorting of the quantized
13 AC coefficients with respect to their magnitudes but rather a search for a specified
14 number “k” of largest magnitude AC coefficients is all that is required. This task can be
15 performed exactly or approximately in different ways so as to avoid the complexity
16 associated with a conventional sorting procedure. In general, a relatively large number of
17 the 63 AC DCT coefficients will have a quantized value of zero. Only the non-zero
18 coefficients need be included in the sorting process. Moreover, if there are “n” non-zero
19 coefficients and only “k” of them having the largest magnitudes are to be preserved in the
20 output stream, then the sorting process may be terminated immediately after only the
21 largest magnitude “k” coefficients have been found, or equivalently immediately after
22 only the smallest magnitude “n-k” coefficients have been found. Moreover, the sorting

1 procedure itself can be different depending on a comparison of "k" to "n" in order to
2 minimize computations.

3 With reference to FIG. 16, there is shown a flowchart of a procedure that selects
4 one of a number of techniques for finding a certain number "k" of largest values out of a
5 set of "n" values. In a first step 271, execution branches to step 272 if "k" is less than $\frac{1}{2}$
6 "n." In step 272, execution branches to step 273 if "k" is much less than $\frac{1}{2}$ "n." In step
7 273, the first "k" values are sorted to produce a list of "k" sorted values, and then the last
8 "n-k" values are scanned for any value greater than the minimum of the sorted "k"
9 values. If a value greater than the minimum of the sorted "k" values is found, then that
10 minimum value is removed and the value greater than the minimum value is inserted into
11 the list of "k" sorted values. At the end of this procedure, the list of sorted "k" values
12 will contain the maximum "k" values out of the original "n" values. A specific example
13 of this procedure is described below with reference to FIG. 17.

14 In step 272, if "k" is not much less than $\frac{1}{2}$ "n", then execution branches to step
15 274. In step 274, a bubble-sort procedure is used, including "k" bottom-up bubble-sort
16 passes over the "n" values to put "k" maximum values on top of a sorting table. An
17 example of such a bubble-sort procedure is listed below:

18
19 /* TABLE(0) to TABLE(n-1) INCLUDES n VALUES */
20 /* MOVE THE k LARGEST OF THE n VALUES IN TABLE TO THE RANGE
21 TABLE(0) TO TABLE(k-1) IN THE TABLE */
22 /* k <= $\frac{1}{2}$ n */
23 FOR i=1 to k


```

1  FOR j=1 to n-i
2  IF (TABLE(n-j) > TABLE(n-j-1)) THEN(
3      /* SWAP TABLE(n-j) WITH TABLE(n-j-1) */
4      TEMP ← TABLE(n-j)
5      TABLE(n-j) ← TABLE(n-j-1)
6      TABLE(n-j-1) ← TEMP
7  NEXT j
8  NEXT I

```

In step 271, if “k” is not less than $\frac{1}{2}$ “n”, then execution branches to step 275. In step 275, if “k” is much greater than $\frac{1}{2}$ “n”, then execution branches to step 276. In step 276, a procedure similar to step 273 is used, except the “n-k” minimum values are maintained in a sorted list, instead of the “k” maximum values. In step 276, the last “n-k” values are placed in the sort list and sorted, and then the first “k” values are scanned for any value less than the maximum value in the sorted list. If a value less than the maximum value in the sorted list is found, then the maximum value in the sorted list is removed, and the value less than this maximum value is inserted into the sorted list. At the end of this procedure, the values in the sorted list are the “n-k” smallest values, and the “k” values excluded from the sorted list are the “k” largest values.

In step 275, if “k” is not much greater than $\frac{1}{2}$ “n”, then execution branches to step 277. In step 277, a bubble-sort procedure is used, including “n-k” top-down bubble-sort passes over the “n” values to put “n-k” minimum values at the bottom of a sorting table.

1 Consequently, the k maximum values will appear in the top “k” entries of the table. An
2 example of such a bubble-sort procedure is listed below:

3
4 /* TABLE(0) to TABLE(n-1) INCLUDES n VALUES */
5 /* MOVE THE n-k SMALLEST OF THE n VALUES IN THE TABLE */
6 /* TO THE RANGE TABLE(k) TO TABLE(n-1) IN THE TABLE */
7 /* $n > k \geq \frac{1}{2} n$ */
8 FOR i=1 to n-k
9 FOR j=0 to n-i-1
10 IF (TABLE(j) < TABLE(j+1)) THEN(
11 /* SWAP TABLE(j) WITH TABLE(j+1)*/
12 TEMP ←TABLE(j)
13 TABLE(j) ← TABLE(j+1)
14 TABLE(j+1) ← TEMP
15 NEXT j
16 NEXT I

17
18 Turning now to FIG. 17, there is shown a flowchart of a procedure for finding up
19 to a specified number “k” of largest magnitude AC DCT coefficients from a set of “n”
20 coefficients, corresponding to the procedure of FIG. 16 for the case of $k \ll \frac{1}{2}n$. In a first
21 step 281, a counter “i” is set to zero. In step 282, the next AC DCT coefficient is
22 obtained from the input stream of original-quality MPEG-2 coded data. If an EOB
23 marker is reached, as tested in step 283, then execution returns. In step 284, the counter

1 "i" is compared to the specified number "k", and if "i" is less than "k", execution
2 continues to step 285. In step 285, a coefficient index and magnitude for the AC DCT
3 coefficient is placed on a sort list. In step 286, the counter "i" is incremented, and
4 execution loops back to step 282.

5 Once the sort list has been loaded with indices and magnitudes for "k" AC DCT
6 coefficients and one additional coefficient has been obtained from the input stream,
7 execution branches from step 284 to step 287. In step 287 the list is sorted by magnitude,
8 so that the minimum magnitude appears at the end of the list. Then in step 288 the
9 coefficient magnitude of the current coefficient last obtained from the input stream is
10 compared to the magnitude at the end of the list. If the coefficient magnitude of the
11 current coefficient is not greater than the magnitude appearing at the end of the list, then
12 execution continues to step 289 to get the next AC DCT coefficient from the input
13 stream. If an EOB marker is reached, as tested in step 290, then execution returns.
14 Otherwise, execution loops back to step 288.

15 In step 288, if the magnitude of the current coefficient is greater than the
16 magnitude at the end of the list, then execution branches to step 291. In step 291, the
17 entry at the end of the list is removed. In step 292, a binary search is performed to
18 determine the rank position of the magnitude of the current coefficient, and in step 293,
19 the current coefficient index and magnitude are inserted into the list at the rank position.
20 The list, for example, is a linked list in the conventional fashion to facilitate the insertion
21 of an entry for the current coefficient at any position in the list. After step 293, execution
22 loops back to step 288.

003222T 99505260

1 An approximation technique of coefficient magnitude classification can be used to
2 reduce the computational burden of sorting by coefficient magnitude. A specific example
3 is the use of hashing of the coefficient magnitude and maintaining lists of the indices of
4 coefficients having the same magnitude classifications. As shown in FIG. 18, a hash
5 table 300 is linked to hash lists 301 storing the indices of classified coefficients. As
6 shown, the hash table 300 is a list of 2^M entries, where "M" is three, and an entry has a
7 value of zero if its associated list is empty, and otherwise the entry has a pointer to the
8 end of the coefficients in its associated list. The lists shown in FIG. 18 have fixed
9 memory allocations in which the pointers in the hash table also indicate the number of
10 coefficient indices in the respective hash lists. Alternatively, the hash lists could be
11 dynamically allocated and linked in the conventional fashion.

12 FIG. 19 shows a flowchart of a procedure for using the hash table 300 and hash
13 lists 301 of FIG. 18 to perform a sort of "k" coefficients having approximately the largest
14 magnitudes from a set of "n" coefficients. This approximation technique ensures that
15 none of the "k" coefficients selected will have a magnitude that differs by more than a
16 certain error limit from the smallest magnitude value of "k" coefficients having the
17 largest magnitude. The error limit is established by the number of hash table entries, and
18 it is the range of the magnitudes that can be hashed to the same hash table entry.

19 In a first step 311 in FIG. 19, the hash table is cleared. Then in step 312, the next
20 AC DCT coefficient is obtained from the input stream. If an EOB marker is not reached,
21 as tested in step 313, then execution continues to step 314. In step 314, a hash table
22 index is stripped from the most significant bits (MSBs) of the coefficient magnitude. For
23 the hash table in FIG. 18 having eight entries, the three most significant bits of the

1 coefficient magnitude are stripped from the coefficient magnitude. This is done by a bit
2 masking operation together with a logical arithmetic shift operation. Then in step 315,
3 the coefficient index is inserted on the hash list of the indexed hash table entry. For
4 example, the hash table entry is indexed to find the pointer to where the coefficient index
5 should be inserted, and then the pointer in the hash table entry is incremented. After step
6 315, execution loops back to step 312. Once all of the AC coefficients for the block have
7 been classified by inserting them in the appropriate hash lists, an EOB marker will be
8 reached, and execution will branch from step 313 to step 316.

9 Beginning in step 316, the hash table and hash lists are scanned to find
10 approximately the "k" largest magnitude coefficients. The hash lists linked to the bottom
11 entries of the hash table will have the indices for the largest magnitude coefficients. Each
12 hash list is scanned from its first entry to its last entry, so that each hash list is accessed as
13 a first-in-first-out queue. Therefore, in each magnitude classification, the coefficient
14 ordering in the output stream will be the same as the coefficient ordering in the input
15 stream, and the approximation will have a "low pass" effect in which possibly some
16 lower-frequency coefficients having slightly smaller magnitudes will be retained at the
17 expense of discarding some higher-frequency coefficients having slightly larger
18 magnitudes. (The approximation results from the fact that the last hash list to be scanned
19 is not itself sorted, and to eliminate the error of the approximation, the last hash list to be
20 scanned could be sorted.)

21 In step 316, a scan index "i" is set to $2^M - 1$ in order to index the hash table
22 beginning at the bottom of the table, and a counter "j" is set equal to "k" in order to stop
23 the scanning process after finding "k" coefficients. Next, in step 317, the hash table is

1 quantization, scanning, and/or (run, level) coding of the coefficients in the original
2 MPEG-2 clip. A study of this problem resulted in a discovery that it is sometimes
3 possible to reduce the number of bits for (run, level) coding of coefficients for an 8x8
4 block including a given number of the non-zero largest magnitude AC DCT coefficients
5 if additional coefficients are also (run, level) coded for the block.

6 The (run, level) coding of the non-zero AC DCT coefficients from the
7 FDSNR_LM procedure has been found to require more bits than from the FDSNR_LP
8 procedure due to an increased occurrence frequency of escape sequences for the (run,
9 level) coding. The increased frequency of escape sequences is an indication that the
10 statistical likelihood of possible (run, level) combinations for the non-zero AC DCT
11 coefficients selected by the FDSNR_LM procedure is different from the statistical
12 likelihood of possible (run, level) combinations for the non-zero AC DCT coefficients
13 produced by the standard MPEG-2 coding process and in particular those selected by the
14 FDSNR_LP procedure.

15 The MPEG-2 coding scheme assigns special symbols to the (run, level)
16 combinations that occur very frequently in ordinary MPEG-2 coded video. The most
17 frequent (run, level) combinations occur for short run lengths (within the range of about
18 0 to 5, where the run length can range from 0 to 63) and relatively low levels (about 1 to
19 10, where the level can range from 1 to 2048). The most frequent of these special
20 symbols are assigned the shortest variable-length code words (VLCs). If a (run, level)
21 combination does not have such a special symbol, then it is coded as an escape sequence
22 including a 6-bit escape sequence header code word followed by a 6-bit run length
23 followed by a 12 bit signed level. An escape sequence requires a much greater number of

1 bits than the special symbols, which have varying lengths depending on their relative
 2 frequency. In particular, the escape sequences each have 24 bits, and the special symbols
 3 have variable-length code words having a maximum of 17 bits.

4 There are two (run, level) VLC tables. The first coding table is designated
 5 TABLE 0, and the second is designated TABLE 1. These tables specify the (run, level)
 6 combinations having special symbols, and the special symbol for each such combination.
 7 For each table, the (run, level) combinations having special symbols, and the range of the
 8 VLC bit lengths of the special symbols, are summarized below:

9
 10 SUMMARY OF PROPERTIES OF DCT COEFFICIENT TABLE ZERO

11 (Table Zero is Table B.14, p. 135 of ISO/IEC 13818-2 1996E)

12

13	<u>Run</u>	<u>Range of Levels</u>	<u>Range of Code Lengths</u>
14	0	1 to 40	2 to 16
15	1	1 to 18	4 to 17
16	2	1 to 5	5 to 14
17	3	1 to 4	6 to 14
18	4	1 to 3	6 to 13
19	5	1 to 3	7 to 14
20	6	1 to 3	7 to 17
21	7	1 to 2	7 to 13
22	8	1 to 2	8 to 13
23	9	1 to 2	8 to 14

1	19	1	13
2	20	1	13
3	21	1	13
4	22	1	14
5	23	1	14
6	24	1	14
7	25	1	14
8	26	1	14
9	27	1	17
10	28	1	17
11	29	1	17
12	30	1	17
13	31	1	17

14

15 The FDSNR_LP procedure selected AC DCT coefficients have (run, level)
16 symbol statistics that are similar to the statistics of ordinary MPEG-2 coded video, and
17 therefore the FDSNR_LP AC DCT coefficients have a similar frequency of occurrence
18 for escape sequences in comparison to the ordinary MPEG-2 coded video. In contrast,
19 the FDSNR_LM procedure selects AC DCT coefficients resulting in (run, level)
20 combinations that are less likely than the combinations for ordinary MPEG-2 coded
21 video. This is due to two reasons. First, the FDSNR_LM procedure selects AC DCT
22 coefficients having the highest levels. Second, the FDSNR_LM procedure introduces
23 higher run lengths due to the elimination of coefficients over the entire range of

1 coefficient indices. The result is a significantly increased rate of occurrence for escape
2 sequences. Escape sequences form the most inefficient mode of coefficient information
3 encoding in MPEG-2 incorporated into the standard so as to cover important but very
4 rarely occurring coefficient information.

5 In order to improve the rate-distortion performance of the scaled-quality MPEG-2
6 coded video from the FDSNR_LM procedure, the non-zero AC DCT coefficients
7 selected by the FDSNR_LM procedure should be quantized, scanned, and/or (run, level)
8 coded in such a way that tends to reduce the frequency of the escape sequences. For
9 example, if the original-quality MPEG-2 coded video was (run, level) coded using
10 TABLE 0, then the largest magnitude coefficients should be re-coded using TABLE 1
11 because TABLE 1 provides shorter length VLCs for some (run, level) combinations
12 having higher run lengths and higher levels. It is also possible that re-coding using the
13 alternate scan method instead of the zig-zag scan method may result in a lower frequency
14 of occurrence for escape sequences. For example, each picture could be (run, level)
15 coded for both zig-zag scanning and alternate scanning, and the scanning method
16 providing the fewest escape sequences, or the least number of bits total, could be selected
17 for the coding of the reduced-quality coded MPEG video.

18 There are two methods having general applicability for reducing the frequency of
19 escape sequences resulting from the FDSNR_LM procedure. The first method is to
20 introduce a non-zero, "non-qualifying" AC DCT coefficient of the 8x8 block into the list
21 of non-zero qualifying AC DCT coefficients to be coded for the block. In this context, a
22 "qualifying" coefficient is one of the k largest magnitude coefficients selected by the
23 FDSNR_LM procedure. The non-qualifying coefficient referred to above, must be lying

1 in between two qualifying AC DCT coefficients (in the coefficient scanning order) that
 2 generate the (run, level) combination causing the escape sequence. Moreover, this non-
 3 qualifying coefficient must cause the escape sequence to be replaced with two shorter
 4 length VLCs when the AC DCT coefficients are (run, level) coded. This first method has
 5 the effect of not only decreasing the number of bits in the coded reduced-quality MPEG
 6 video in most cases, but also increasing the PSNR.

7 The qualifying AC DCT coefficient causing the escape sequence that is first in the
 8 coefficient scanning order will be simply referred to as the first qualifying coefficient.
 9 The qualifying AC DCT coefficient causing the escape sequence that is second in the
 10 coefficient scanning order will be simply referred to as the second qualifying coefficient.
 11 For example, suppose the qualifying coefficients in zig-zag scan order for an 8x8 block
 12 include C_{51} followed by C_{15} having a level of 40. If only the qualifying coefficients were
 13 (run, level) coded for the microblock, C_{15} would result in a run length of 3, because there
 14 are a total of three non-qualifying coefficients (C_{42} , C_{33} , and C_{24}) between C_{51} and C_{15} in
 15 the scan order. Therefore, C_{15} would have to be coded as an escape sequence, because a
 16 run of 3 and level of 40 does not have a special symbol. In this example, the escape
 17 sequence is in effect caused by a first qualifying coefficient, which is C_{51} , and a second
 18 qualifying coefficient, which is C_{15} . This escape sequence can possibly be eliminated
 19 say, if C_{24} is a non-zero, non-qualifying coefficient of the block, C_{24} has a level of 5 or
 20 less, and C_{24} is (run, level) coded together with the qualifying coefficients. For example,
 21 assuming that C_{24} has a level of 5, and using the MPEG-2 (run, level) coding TABLE 1,
 22 then C_{24} has a run length of two and is coded as the special symbol 0000 0000 1010 0s,
 23 where "s" is a sign bit, and C_{15} now has a run length of 0 and is coded as the special

1 symbol 0000 0000 0010 00s. Such a consideration clearly applies to the rest of the non-
2 zero non-qualifying coefficients lying in between the two qualifying coefficients
3 producing the escape sequence. In the above example, these non-qualifying coefficients
4 are C_{42} and C_{33} .

5 Whether or not an escape sequence can be eliminated from the (run, level) coding
6 of the qualifying coefficients can be determined by testing a sequence of conditions. The
7 first condition is that the second qualifying coefficient must have a level that is not
8 greater than the maximum level of 40 for the special (run, level) symbols. If this
9 condition is satisfied, then there must be a non-zero non-qualifying AC DCT coefficient
10 that is between the first and second qualifying coefficients in the coefficient scanning
11 order. If there is such a non-qualifying coefficient, then the combination of its level and
12 the run length between the first qualifying coefficient and the non-qualifying coefficient
13 in the coefficient scanning order must be one of the special (run, level) symbols. If so,
14 then the combination of the level of the second qualifying coefficient and the run length
15 between the non-qualifying coefficient and the second qualifying coefficient must also be
16 a special (run, level) symbol, and if so, all required conditions have been satisfied. If not,
17 then the conditions with respect to the non-qualifying coefficient are successively applied
18 to any other non-zero non-qualifying AC DCT coefficient of the block lying in between
19 the two qualifying coefficients, until either all conditions are found to be satisfied or all
20 such non-qualifying coefficients are tested and failed. If there are sufficient
21 computational resources, this search procedure should be continued to find all such non-
22 qualifying coefficients that would eliminate the escape sequence, and to select the non-

1 sequence results, as tested in step 333, or until the end of the block is reached, as tested in
2 step 336.

3 With reference to FIG. 21, there is shown a flow chart of the subroutine (that was
4 called in step 335 of FIG. 20) for attempting to find a non-zero, non-qualifying AC DCT
5 coefficient that can be (run, level) coded to eliminate an escape sequence for a qualifying
6 coefficient. In a first step 341, the procedure identifies the first qualifying coefficient and
7 the second qualifying coefficient causing the escape sequence. For example, the
8 subroutine of FIG. 21 can be programmed as a function having, as parameters, a pointer
9 to a list of the non-zero AC DCT coefficients in the scan order, an index to the first
10 qualifying coefficient in the list, and an index to the second qualifying coefficient in the
11 list. In step 342, the subroutine looks for a non-zero non-qualifying AC DCT coefficient
12 between the first and the second qualifying coefficients in the scan order. For example,
13 the value of the index to the first qualifying coefficient is incremented and compared to
14 the value of the index for the second qualifying coefficient, and if they are the same, there
15 is no such non-qualifying coefficient. Otherwise, if the new coefficient pointed to (by
16 incrementing the index of the first qualifying coefficient) is a non-zero coefficient then it
17 becomes a candidate non-qualifying coefficient deserving further testing. If however the
18 new coefficient pointed to (by incrementing the index of the first qualifying coefficient)
19 has a value zero then it is not a candidate non-qualifying coefficient. If no such
20 (candidate) non-qualifying coefficients are found, as tested in step 343, then execution
21 returns from the subroutine with a return code indicating that the search has been
22 unsuccessful. Otherwise, execution continues to step 344.

1 In step 344, the non-qualifying coefficient is (run, level) coded, to determine in
2 step 345 whether it codes to an escape sequence. If it codes to an escape sequence, then
3 execution loops back from step 345 to step 342 to look for another non-zero non-
4 qualifying AC DCT coefficient in the scan order between the first and second qualifying
5 coefficients. If it does not code to an escape sequence, then execution continues from
6 step 345 to step 346. In step 346, the second qualifying coefficient is (run, level) coded,
7 using the new run length, which is the number of coefficients in the scan order between
8 the non-qualifying coefficient and the second qualifying coefficient. If it codes to an
9 escape sequence, as tested in step 347, then execution loops back from step 347 to step
10 342 to look for another non-zero non-qualifying AC DCT coefficient in the scan order
11 between the first and second qualifying coefficients. If it does not code to an escape
12 sequence, then execution continues from step 347 to step 348.

13 In step 348, execution returns with a successful search result unless a continue
14 search option has been selected. If the continue search option has been selected, then
15 execution branches from step 348 to step 349 to search for additional non-zero non-
16 qualifying AC DCT coefficients that would eliminate the escape sequence. In other
17 words, steps 342 to 347 are repeated in an attempt to find additional non-zero non-
18 qualifying AC DCT coefficients that would eliminate the escape sequence. If no more
19 such non-qualifying coefficients are found, as tested in step 350, execution returns with a
20 successful search result. Otherwise, execution branches from step 350 to step 351 to
21 select the non-qualifying coefficient giving the shortest overall code word length and/or
22 the largest magnitude for the best PSNR, and execution returns with a successful search
23 result. For example, for each non-qualifying coefficient that would eliminate the escape

1 sequence, the total bit length is computed for the (run, level) coding of the non-qualifying
2 coefficient and the second qualifying coefficient. Then a search is made for the non-
3 qualifying coefficient producing the shortest total bit length, and if two non-qualifying
4 coefficients which produce the same total bit length are found, then the one having the
5 largest level is selected for the elimination of the escape sequence.

6 A second method of reducing the frequency of occurrence of the escape
7 sequences in the (run, level) coding of largest magnitude AC DCT coefficients for an 8x8
8 block is to change the mapping of coefficient magnitudes to the levels so as to reduce the
9 levels. Reduction of the levels increases the likelihood that the (run, level) combinations
10 will have special symbols and therefore will not generate escape sequences. This second
11 method has the potential of achieving a greater reduction in bit rate than the first method,
12 because each escape sequence can now be replaced by the codeword for one special
13 symbol, rather than by the two codewords as is the case for the first method. The second
14 method, however, may reduce the PSNR due to increased quantization noise resulting
15 from the process producing the lower levels. Therefore, if a desired reduction of escape
16 sequences can be achieved using the first method, then there is no need to perform the
17 second method, which is likely to reduce the PSNR. If the first method is used but not all
18 of the escape sequences have been eliminated, then the second method could be used to
19 possibly eliminate the remaining escape sequences.

20 The mapping of coefficient magnitudes to the levels can be changed by decoding
21 the levels to coefficient magnitudes, changing the quantization scale factor (q_{si}), and then
22 re-coding the levels in accordance with the new quantization scale factor (q_{si}). The
23 quantization scale factor is initialized in each slice header and can also be updated in the

1 macroblock header on a macroblock basis. Therefore it is a constant for all blocks in the
 2 same macroblock. In particular, the quantization scale factor is a function of a
 3 q_scale_type parameter and a $quantizer_scale_code$ parameter. If $q_scale_type = 0$, then
 4 the quantizer scale factor (qsi) is twice the value of q_scale_code . If $q_scale_type = 1$,
 5 then the quantizer scale factor (qsi) is given by the following table, which is the right half
 6 of Table 7-6 on page 70 of ISO/IEC 13838-2:1996(E):

8	<u>quantizer scale code</u>	<u>quantization scale factor (qsi)</u>
9	1	1
10	2	2
11	3	3
12	4	4
13	5	5
14	6	6
15	7	7
16	8	8
17	9	10
18	10	12
19	11	14
20	12	16
21	13	18
22	14	20
23	15	22

1	20	27
2	21	28
3	22	29
4	23	30
5	24	31

6

7 In a preferred method for generation of trick mode files, the quantization scale

8 factor is adjusted in order to achieve a desired reduction in the escape sequence

9 occurrence frequency resulting from the modified FDSNR_LM procedure, and the

10 number (k) of largest magnitude coefficients is adjusted in order to achieve a desired

11 reduction in bit rate. A specific implementation is shown in the flow chart of FIGS. 22-

12 23. In a first step 361, the number (k) of largest magnitude AC coefficients per 8x8 block

13 is initially set to a value of 9, and the quantization scaling factor (QSF) is initially set to a

14 value of 2. Then conversion of the I frames of an original-quality MPEG-2 coded video

15 clip to a lower quality level begins. When a picture header is encountered in step 362,

16 indicating the beginning of a new I frame, execution continues to step 363. In step 363,

17 execution branches depending on the value of the intra_vlc_format parameter in the

18 picture header of the original-quality MPEG-2 coded video clip. This value is either 0,

19 indicating that the first (run, level) coding table (TABLE 0) was used for coding the

20 picture, or 1, indicating that the second (run, level) coding table (TABLE 1) was used for

21 coding the picture. In either case, the down scaled quality picture will be coded with the

22 second (run, level) coding table. If the intra_vlc_format parameter is equal to 0 execution

23 continues from step 363 to step 364 where TABLE 0 is read in for (run, level) symbol

1 decoding in the original-quality MPEG-2 coded clip. Otherwise, if the intra_vlc_format
2 parameter is equal to 1, then execution continues from step 363 to step 365 where
3 TABLE 1 is read in for (run, level) symbol decoding in the original-quality MPEG-2
4 coded clip.

5 After steps 364 and 365, execution continues to step 366. In step 366, the
6 modified FDSNRS_LM procedure is applied to the 8x8 blocks of the current slice, using
7 the adjusted quantization scale index, if the adjusted quantization scale index is less than
8 the maximum possible quantization scale index. In step 367, execution loops back to step
9 362 to continue 8x8 block conversion until a new slice header is encountered, indicating
10 the beginning of a new slice. Once a new slice is encountered, execution continues from
11 step 367 to step 368. In step 368, the average escape sequence occurrence frequency per
12 block for the last slice is compared to a threshold TH1. If the escape sequence
13 occurrence frequency is greater than the threshold, then execution branches to step 369.
14 In step 369, if the quantization scaling factor (QSF) is less than or equal to a limit value
15 such as 2, then execution branches to step 370 to increase the quantization scaling factor
16 (QSF) by a factor of two.

17 In step 368, if the escape sequence occurrence frequency is not greater than the
18 threshold TH1, then execution continues to step 371 of FIG. 23. In step 371, the average
19 escape sequence occurrence frequency per 8x8 block for the last slice is compared to a
20 threshold TH2. If the escape sequence occurrence frequency is less than the threshold
21 TH2, then execution branches to step 372. In step 372, if the quantization scaling factor
22 (QSF) is greater than or equal to a limit value such as 2, then execution branches to step
23 373 to decrease the quantization scaling factor (QSF) by a factor of two. After step 373,

1 and also after step 370 of FIG. 22, execution continues to step 374 of FIG. 23. In step
2 374, execution continues to step 375 if a backtrack option has been selected. In step 375,
3 re-coding for the last slice is attempted using the adjusted quantization scale factor. The
4 new coding, or the coding that gives the best results in terms of the desired reduction of
5 escape sequence occurrence frequency, is selected for use in the scaled quality picture.
6 After step 375, execution continues to step 376. Execution also continues to step 376
7 from: step 369 in FIG 22 if the quantization scaling factor (QSF) is not less than or equal
8 to 2; step 371 in FIG 23 if the escape sequence occurrence frequency is not less than the
9 threshold TH2; step 372 in FIG 23 if the quantization scaling factor (QSF) is not greater
10 than or equal to 2; and from step 374 in FIG 23 if the backtrack option has not been
11 selected.

12 In step 376, the average bit rate of the (run, level) coding per 8x8 block for at
13 least the last slice is compared to a high threshold TH3. Preferably this average bit rate is
14 a running average over the already processed segment of the current scaled quality I-
15 frame, and the high threshold TH3 is selected to prevent video buffer overflow in
16 accordance with the MPEG-2 Video Buffer Verifier restrictions. If the average bit rate
17 exceeds the high threshold TH3, then execution continues to step 377, where the number
18 (k) of non-zero largest magnitude AC coefficients per 8x8 block is compared to a lower
19 limit value such as 6. If the number (k) is greater than or equal to 6, then execution
20 continues to step 378 to decrement the number (k).

21 In step 376, if the average bit rate is not greater than the threshold TH3, then
22 execution continues to step 379. In step 379, the average bit rate is compared to a lower
23 threshold TH4. If the average bit rate is less than the threshold TH4, then execution

branches from step 379 to step 380, where the number (k) of non-zero largest magnitude AC DCT coefficients per 8x8 block is compared to a limit value of 13. If the number (k) is less than or equal to 13, then execution continues to step 381 to increment the number (k). After step 378 or 381, execution continues to step 382. In step 382, execution continues to step 383 if a backtrack option is selected. In step 383, an attempt is made to re-code the last slice for the scaled quality picture using the adjusted value of the number (k) of non-zero largest magnitude AC DCT coefficients per block. After step 383, execution loops back to step 362 of FIG. 22 to continue generation of the scaled quality clip. Execution also loops back to step 362 of FIG. 22 after: step 377 if the value of (k) is not greater than or equal to 6; step 379 if the average bit rate is not less than the threshold TH4; step 380 if the value of (k) is not less than or equal to 13; and step 382 if the backtrack option has not been selected. Coding of the scaled quality clip continues until the end of the original quality clip is reached in step 364 of FIG. 22, in which case execution returns.

15 In a preferred implementation, a fast forward trick mode file and a fast reverse
16 trick mode file are produced from an original-quality MPEG-2 coded video main file
17 when the main file is ingested into the video file server. As shown in FIG. 24, a volume
18 generally designated 390 is allocated to store the main file 391. The volume 390 includes
19 an allocated amount of storage that exceeds the real file size of the main file 391 in order
20 to provide additional storage for meta-data 392, the fast forward trick file 393, and the
21 fast reverse trick file 394. The trick files are not directly accessible to clients as files;
22 instead, the clients may access them thorough trick-mode video service functions. With

1 files. For example, a transition from the main file to one of the trick files will usually
2 involve a discontinuity in the mean video decoder main buffer fullness level, because
3 only the I frames of the main file correspond to frames in the trick files, and the
4 corresponding I frames have different bit rates when the trick mode I frames are scaled
5 down for a reduced bit rate. An instantaneous transition from a trick file back to the main
6 file may also involve a discontinuity especially when freeze frames are inserted between
7 the I frames for trick mode operation. To avoid these discontinuities, the seamless
8 splicing procedure of FIGS. 3 to 6 as described above is used during the transitions from
9 regular play mode into trick mode and similarly from trick mode back into the regular
10 play mode. Through the use of the seamless splicing procedure to modify the video
11 stream content, for example for the "Seamless Splice" locations identified in FIG. 26A,
12 the video decoder main buffer level will be managed so as to avoid both overflows and
13 underflows leading to visual artifacts.

14 It is desired to copy in and out of the volume 390 with or without the meta-data
15 392 and the trick files 393, 394. This is useful to export and/or import complete files
16 without regenerating the trick files. The file encoding type is now recognized as a part of
17 the volume name. Therefore there can be multiple kinds of access to these files. The
18 read and write operations are done by derivations of the class file system input/output
19 (FSIO) which takes into account the proper block offset of the data to read or write.
20 There is one derivation of FSIO per encoding type, providing three different access
21 modes. EMGP3, MPEG2, and RAW. EMPEG2 accesses the whole volume from the
22 beginning of the meta-data array, and in fact provides access to the entire volume except
23 the inode 401, but no processing is done. MPEG2 access only the main part of the asset

1 only modification done is an update of the video PTS, which must be continuous. Then,
 2 the GOP index is written on disk. This avoids reading again the file while generating the
 3 second trick file. The GOP index size is: 24 times the GOP number. In the worst case
 4 (the file is assumed not to be 1 frame only), there are 2 frames per GOP and 30 frames
 5 per second. So for 1 hour in fast forward, the GOP index size is: $(24 \times 3600 \times 30) / 2 =$
 6 1296000 bytes. This will be the case for a 4 hour film played at 4 times the normal
 7 speed. Therefore, this GOP index can be kept in memory during the trick file generations
 8 without risk of memory overflow.

9 The read and write rate are controlled to conserve bandwidth on the cached disk
 10 array. The bandwidth reserved for these generations is a parameter given by the video
 11 service. It is a global bandwidth for both read and writes. The number of disk I/O per
 12 seconds is counted so as not to exceed this bandwidth.

13 The trick files header update is done once when both the fast forward and fast
 14 reverse trick files and the GOP index have been successfully written.

15 Playing a file is done with the CM_MpegPlayStream class. Fast forward
 16 (reverse) can only be requested when we are in the paused state. The current frame on
 17 which we are paused is known from the MpegPause class. This frame is located in the
 18 GOP index of the trick file. Then the clip start point and length are modified in the Clip
 19 instance with the trick file position computed from the beginning of the clip. So, the Clip
 20 class handle these trick files in a manner similar to the main file. The current logical
 21 block number is updated with the block address in the trick file recomputed from the
 22 beginning of the main clip. In fact, a seek is performed in the trick file as it was part of
 23 the main file, which is totally transparent for the ClipList and Clip classes. The transition

1 When there is a transition from play to pause, the only latency issue is related to
2 the buffer queue handled by the player and to the GOP size. The stream can build
3 immediately the active pause GOP, and then this GOP will be sent at the end of the
4 current GOP with a splicing between these two streams.

5 When there are transitions from pause to regular play or fast forward and fast
6 reverse, a seek in the file is done. This means that the current buffer pool content is
7 invalidated and the buffer pool is filled again. Play can start again while the buffer pool
8 is not completely full, as soon as the first buffer is read. The buffer pool prefilling can
9 continue as a background process. The issue here is that there is a risk to generate an
10 extra load on the cached disk array as well as on the stream server side when the buffer
11 pool is being prefilled.

12 To avoid too frequent transitions from play to fast forward and fast reverse, there
13 is a limitation of the number of requests per second for each stream. This limitation is
14 part of the management of the video access commands. A minimum delay between two
15 commands is defined as a parameter. If the delay between a request and the previous one
16 is too small, the request is delayed. If a new request is received during this delay, the
17 new request replaces the waiting one. So the last received request is always executed.

18 The volume parameter (vsparams) file contains these new parameters for the trick
19 mode files:

20 TrickFileExtensionSize:<percent>:

21 DefaultFastAcceleration:<acceleration>:

1 In a preferred implementation of the video service software, a new encoding type
2 is created. The encoding type enum becomes:

```
3 enum encoding-t{  
4     ENC_UNKNOWN    = 0,          /* unknown format */  
5     ENC_RAW        = 1,          /* uninterpreted data */  
6     ENC_MPEG1      = 2,          /* constrained MPEG1 */  
7     EMC_MPEG       = 3,          /* generic MPEG */  
8     ENC_EMPEG2     = 4,          /* MPEG2 with trick files extension */  
9 };
```

10
11 - The encoding information accessible by VCMP_EXTENDEDINFO includes
12 information about trick files:

```
13  
14 struct trickFilesInfo_t{  
15     ulong_t      generationDate;  /* date/time of the generation of the trick  
16         files */  
17     rate_factor_t acceleration;    /* acceleration factor */  
18     ulong_t      framesNumber;    /* frames number in each trick file (FWD and  
19         REV) */  
20     ulong_t      gopNumber;       /* GOP number of each file */  
21 };  
22
```

```
23 struct EMPEG2info_t{
```



```

1      bool_t      overwriteIfExists;
2      rate_factor_t acceleration;
3  };
4
5  VCMPtrickgenres_t      VCMP_TRICKFILESGEN (VCMPtrickfilesargs_t)  =
6  36,

```

8 If the trick files already exist and if the boolean overwriteIfExists is true, then the
 9 trick files are generated again, in the other case nothing is done. Acceleration is the
 10 acceleration as defined and used for the controlled speed play function. It is a percentage
 11 of the normal speed, it must be greater than 200 and smaller than 2000. The special value
 12 of 0 can be used to generate files with the default acceleration defined in vssiteparams.
 13 The procedure starts the generation process. The completion is notified by a callback.

14 The video service includes a new option to copyin and copyout. The option is
 15 added to allow a user to copy all the file or the main asset only. For compatibility with
 16 old client programs, the following new procedures are added:

```

18  VCMPcopyres_t      VCMP_FULL_COPYIN      (copyinargs2_t)      = 37,
19  VCMPcopyres_t      VCMP_FULL_COPYOUT     (copyoutargs2_t)     = 38,

```

21 These new procedures take the same interface as the existing one, but are used to copy-in
 22 the complete file: meta-data + Asset + trick files.

23

1 The video service includes a new procedure

2 VCMP_TRICKFILESGENCOMPLETED, which uses the following structures:

3

4 struct VCMPtrickfilescomplete_t{

5 tHandle_t handle;

6 VCMPstatus_t status;

7 };

8

9 VCMPstatus_t TRICKFILESGENCOMPLETED (VCMPtrickfilescomplete_t) = 10,

11 The video service includes new procedures are added for handling trick mode

12 generation arguments, which uses the following structures:

13

14 struct cms_trick_gen_args {

15 Handle_t Vshandle;

16 name_t name;

17 bool_t overwriteIfExists;

18 rate_factor_t acceleration;

19 bandwidth_t reservedBw;

20 };

21

22 cms_status CMSPROC_GEN_TRICK_FILES (cms_trick_gen_args) = 34,

```

1 struct trick_gen_completed_args {
2     Handle_t      Vshandle;
3     cms_status    status;
4 };
5 void CTLPROC_TRICKGENCOMPLETED (trick_gen_completed_args) = 8,
6

```

7 The video service includes the following option to force the regeneration of trick
8 files even if they exist:

```
9 nms_content -gentrick <name> [<-f>] [acceleration]
```

10 Without this option, an error code is returned if the trick files exist. "Acceleration" is an
11 acceleration factor. If it is not present, the default value is taken in vsparams.

12 The video services include a encoding information access function (nms_content
13 -m). This function produces a displayed output containing, for each trick file generated,
14 the acceleration, the generation date and time, the frames number, and the GOP number.

15 For the use of an FTP copy function with the trick files, the following new
16 commands are added:

```

17
18 nms_content -copyinfull <same arguments as -copyin>
19 nms_content -copyoutfull <same arguments as -copyout>
20

```

21 Another application of the SNR scaling of the invention is to reduce the bit rate of
22 an MPEG-2 transport stream in order to allow combining multiple MPEG-2 transport
23 streams to match a target bit rate for a multiple program transport stream. For example,

1 FIG. 29 shows a system for combining an MPEG-2 audio-visual transport stream 411
2 with an MPEG-2 closed-captioning transport stream 412 to produce a multiplexed
3 MPEG-2 transport stream 413. In this case, the closed captioning transport stream 412,
4 containing alphanumeric characters and some control data instead of audio-visual
5 information, has a very low bit rate compared to the audio-visual transport stream 411.
6 Assuming that the target bit rate for the multiplexed transport stream 413 is the same as
7 the bit rate of the audio-visual transport stream 411, there need be only a slight decrease
8 in the bit rate of the audio-visual transport stream, and this slight decrease can be
9 obtained by occasionally removing one non-zero AC DCT coefficient per 8x8 block.
10 Therefore, in the system of FIG. 29, the audio-visual transport stream 411 is processed by
11 a program module 414 for selective elimination of non-zero AC DCT coefficients to
12 slightly reduce the average bit rate of this transport stream. A transport stream
13 multiplexer 415 then combines the modified audio-visual transport stream with the closed
14 captioning transport stream 412 to produce the multiplexed MPEG-2 transport stream
15 413.

16 In order to determine whether or not any non-zero AC DCT coefficient should be
17 eliminated from a next 8x8 block in the audio-visual transport stream 411, a module 421
18 is executed periodically to compute a desired bit rate change in the audio-visual transport
19 stream 411. For example, respective bit rate monitors 416, 417 may measure the actual
20 bit rate of the audio-visual transport stream 411 and the closed captioning transport
21 stream 412. Alternatively, if it is known precisely how these transport streams are
22 generated, presumed values for the bit rates of these transport streams may be used in lieu
23 of measured bit rates. The computation of the desired bit rate change also includes the

1 desired bit rate 418 for the multiplexed MPEG-2 transport stream, and a bit rate 419 of
2 multiplexer overhead, representing any net increase in bit rate related to the multiplexing
3 of the audio-visual transport stream 411 with the closed captioning transport stream 412.
4 An adder/subtractor 420 combines the various bit rate values from the inputs 416, 417,
5 418, and 419 to compute the desired bit rate change in the audio-visual transport stream
6 411. From the adder/subtractor 420, the module 421 converts the desired change in bit
7 rate to a desired number of bits to be removed per computational cycle (e.g., per
8 millisecond). This number of bits to be removed per computational cycle is received in
9 an adder/subtractor 422, and the output of the adder/subtractor is received in an integrator
10 423. A limiter 424 takes the sign (positive or negative) of the integrated value to produce
11 a flag indicating whether or not one non-zero AC DCT coefficient should be removed
12 from the coefficients for the next 8x8 block, assuming that the next block has at least one
13 non-zero AC DCT coefficient. (Alternatively, a non-zero AC DCT coefficient could be
14 removed only if the 8x8 block has more than a predetermined fraction of the average
15 number of AC DCT coefficients per 8x8 block.) The particular non-zero AC DCT
16 coefficient to remove in each case can be selected using any of the methods discussed
17 above with reference to FIGS. 14, 15, or FIG. 20. For example, the coefficient to remove
18 could be the last non-zero AC DCT coefficient in the scan order. Alternatively, the non-
19 zero AC DCT coefficient having the smallest magnitude could be removed so long as its
20 removal does not cause an escape sequence.

21 When the module 414 removes a non-zero AC DCT coefficient from a 8x8 block,
22 it sends the number of bits removed to the adder/subtractor 422. In a preferred
23 implementation, the operations of the adder/subtractor 422, integrator 423, and limiter

1 424 are performed by a subroutine having a variable representing the integrated value.
2 During each computational cycle, the variable is incremented by the number of bits to be
3 removed per computational interval, and whenever the module 414 removes a non-zero
4 AC DCT coefficient from a 8x8 block of the audio-visual transport stream, the variable is
5 decremented by the number of bits removed.

6 Although the system in FIG. 29 has been described for achieving a slight
7 reduction in bit rate of the MPEG-2 audio-visual transport stream 411 for combining
8 multiple transport stream to produce a multiplexed MPEG-2 transport stream, it should
9 be apparent that it could be used for obtaining relatively large reductions in bit rate. In
10 this case, the module 414 would use the procedure of FIGS. 14, 15 or preferably FIG. 20,
11 and a multi-level comparator 424 would be used instead of a single-level comparator 424.
12 The multi-level comparator would determine a desired number of non-zero coefficients to
13 discard per 8x8 block based on the value of the output of the integrator 423. The
14 maximum number of non-zero AC coefficients to keep for each 8x8 block (i.e., the value
15 of the parameter "k"), for example, would be determined by subtracting the number of
16 non-zero AC DCT coefficients in the 8x8 block from the desired number to discard, and
17 limiting this difference to no less than a predetermined fraction of the average number of
18 non-zero AC coefficients per 8x8 block.

19 In order to produce reduced-quality MPEG-2 data having a bit rate that is a small
20 fraction of the bit rate of the original-quality MPEG-2 data, the bits for each frame should
21 be allocated to each DCT coded block (i.e., an 8x8 pixel block) in the frame in such a
22 way that more complex blocks are allocated more bits than simple blocks. With a
23 relatively large reduction in bit rate, there will be degradation in picture quality, and the

1 same offset S of bits in the reduced rate MPEG-2 data that are not coefficient bits.
2 Therefore, if the original-quality MPEG-2 data has a total bit rate of BO, a padding PD
3 and offset S, and it is desired for the reduced-quality MPEG-2 data to have a total bit rate
4 of BR, then the coefficient bit reduction factor RF is $RF = (BR-S)/(BO-PD-S)$. BR-S
5 represents the video bit rate of the reduced bit rate file and BO-PD-S represents the video
6 bit rate of the original quality MPEG-2 data. In many cases the offset S will be relatively
7 small compared to BR and BO so that any change or uncertainty in the value of the offset
8 has a small effect on the value of the reduction factor. Therefore, the offset S could be
9 estimated based on frame size and frame type (e.g., I, P or B) depending on the desired
10 uncertainty in the value of the reduction factor RF. S is always greater than zero.

11 In general, the number of bits allocated to the non-zero AC DCT coefficients of
12 an 8x8 block in a frame in the reduced-quality MPEG-2 data will be less than the number
13 of bits allocated to the non-zero AC DCT coefficients of the corresponding 8x8 block of
14 the same frame in the original-quality MPEG-2 data. This is due to the fact that the
15 preferred process of bit rate reduction is eliminating non-zero AC DCT coefficients.
16 Therefore, the number of bits allocated for the 8x8 block is reduced by a variable number
17 each time a coefficient is eliminated. It is possible although not likely that the dropping
18 of one or more coefficients will cause the number of bits allocated to the 8x8 block to
19 become equal to the number of bits that are available for encoding the non-zero AC DCT
20 coefficients for the reduced 8x8 block. It is most likely that some of the bits available for
21 encoding the non-zero AC DCT coefficients for the reduced 8x8 block will not be used
22 for encoding the non-zero AC DCT coefficients for the reduced 8x8 block. So that the
23 actual bit rate for the reduced-quality MPEG-2 data will not become significantly less

1 and 8 columns of 8x8 blocks, although in practice, the frame would usually have a larger
2 number of rows and columns of 8x8 blocks. Inside the cells of the matrix of the frame
3 501 are compressed 8x8 blocks, such as the compressed block 502. The compressed
4 blocks are shown having a varying size. The size of each compressed block represents
5 the total number of bits used in the variable-length encoding of the non-zero AC DCT
6 coefficients in the block.

7 Shown below the frame 501 is a corresponding reduced size frame 503
8 representing a frame in the reduced-quality MPEG coded video produced from the
9 original-quality MPEG coded video. In particular, the reduced size frame 503 is a matrix
10 having the same number of rows and columns of 8x8 blocks. Each 8x8 block in the
11 reduced frame 503 is produced from a corresponding 8x8 block in the original frame 501
12 by a process of selective removal of non-zero AC DCT coefficients so that the 8x8 block
13 in the reduced frame 504 has a proportionately reduced number of bits encoding non-zero
14 AC DCT coefficients, subject to smoothing of deviations among the reduced 8x8 blocks.
15 For example, the reduced 8x8 block 504 is produced by selective removal of non-zero
16 AC DCT coefficients from the compressed block 502.

17 It is desired to keep the ratio between each original block size and the reduced
18 rate block size constant and equal to the ratio of the original frame size and the reduced
19 bit rate frame size. This is not always possible because the bit allocation per block differ
20 from frame to frame due to local simple blocks and a variable proportion of complex sub
21 frames in a frame. For example, there are blocks for which the maximum number if bits
22 is smaller than the average block size computed as:

23

1 Breduced(j) = Boriginal(j)/MAoriginal(n) * BRreduced

2

3 Where (j) is the index of the block in the frame, (n) is the present frame number,

4 Breduced(j) is the number of bits allocated for the reduced bit rate block (j) and it is

5 proportional to the original block size Boriginal(j), MAreduced(n) is the moving average

6 of original frame size after (n) frames and BRreduced is the target reduced video bit rate.

7 The average block size of the reduced bit rate frame is calculated as:

8

9 $AVB_{reduced} = F_{reduced}/NB$

10

11 Where $F_{reduced}$ is the frame size of the reduced-quality file, and NB is the number of

12 blocks per frame and it is calculated to conform to the MPEG-2 standard from the frame

13 resolution as follows:

14

15 $NB = \text{ceil}(FR(\text{horizontal})/8) * \text{ceil}(FR(\text{vertical})/8)$

16

17 where "ceil" is the conversion to a rounded-up integer, $FR(\text{horizontal})$ is the horizontal

18 frame resolution in pixels (720 for NTSC and 528 for PAL), and $FR(\text{vertical})$ is the

19 vertical frame resolution in pixels (480 for NTSC and PAL).

20 In other cases the number of bits is greater than the average size allocated as

21 shown in the reduced frame 503. It should also be apparent from the above discussion

22 that sometimes all of the bits allocated to the reduced frame cannot be used. Bits that are

1 allocated but not used in the reduced frame should be distributed by a smoothing
2 mechanism to another reduced frame that may be more complex.

3 Fig 30 shows how some reduced blocks are smaller than a proportionally-reduced
4 size and the bucket is filled, and some reduced blocks are larger than the proportionally-
5 reduced size and the bucket is emptied. This methodology is especially efficient in cases
6 when there are very large variations between adjoining blocks based on the details of the
7 block, for example hair strings near a talking head. For example, the reduced frame 503
8 includes a "bucket filling" block 505 and a "bucket emptying" block 507. A rectangle
9 506 around the bucket filling block 505 indicates a proportional reduction in the number
10 of bits of the non-zero AC DCT coefficients in the corresponding compressed block 509
11 in the original frame 501. The difference in area between the outer rectangle 506 and the
12 bucket filling block 505 indicate bits that are available for encoding non-zero AC DCT
13 coefficients of the bucket filling block 505 but were not used and instead were placed in
14 the bucket. A rectangle 508 inside the bucket emptying block 507 indicates a
15 proportional reduction in the number of bits of the non-zero AC DCT coefficients in the
16 corresponding compressed block 510 in the original frame 501. The difference in area
17 between the inner rectangle 508 and the bucket emptying block 507 represents bits that
18 were taken from the bucket and allocated to the bucket emptying block 507.

19 When a block of a frame is simple, the "bucket" will fill, and when a block of the
20 frame is very complex the "bucket" will empty. All this is done by including as many
21 coefficients for a block as possible given a target size of bits in proportion to the non-zero
22 AC DCT coefficient bits in the corresponding block of the original encoded frame 501,

1 plus a share of any bits in the bucket. In general, the number (m) of non-zero AC DCT
2 coefficients to keep for the block is found such that:

3

$$4 \quad \text{ABS}(\text{Sum}(\text{Bits}(\text{DCT}(1))+\dots+\text{Bits}(\text{DCT}(m)) - \text{Breduced}(j)) < \text{Bucket}/\text{NBR}(j)$$

5

6 Where Bucket is the number of bits available in the "bucket" (size of the "bucket"), and
7 NBR(j) is the remaining number of blocks in the frame including block (j), calculated as
8 NB-j, so that the share of any bits in the bucket is Bucket/NBR(j). The DCT coefficients
9 are obtained by parsing the original encoded block. The non-zero AC DCT coefficients
10 that are retained in the reduced frame could be selected in the order of parsing, or in the
11 order of the largest magnitude coefficients. In any case, the "bucket" remains as empty
12 as possible considering that excess bits from one block will be shared with at least a
13 certain number (NB) of following blocks.

14 The use of the bucket as a smoothing mechanism with a frame and between
15 frames provides a significant improvement in picture quality because there is a large
16 variability between the number of bits per DCT coefficient due to the MPEG-2 encoding
17 code books and the number of escape sequences. For example, suppose Bucket = 36000
18 bits, NB=48*30=1440, SUM(DCT(1) + ... + DCT(11)) = 200, SUM(DCT(1) + ... +
19 DCT(12)) = 300, and Breduced(j) = 280. The formula above will result in:

$$20 \quad \text{ABS}(200-280) = 80 > 36000/1440 = 25$$

$$21 \quad \text{ABS}(300-280) = 20 < 36000/1440 = 25$$

22

1 So we keep 12 DCT coefficients. In some cases there are only 11 DCT coefficients in the
2 example and there is no choice but to fill the "bucket" with 80 more bits.

3 FIG. 31 shows a graph of frame size (i.e., number of bits per frame) as a function
4 of frame number for various traces. A first trace 521 is for the original-quality MPEG
5 coded video, and a second trace 522 is for the reduced-quality MPEG coded video
6 produced from the original-quality MPEG coded video using the adaptive bit rate
7 reduction method as introduced above and as further described below. FIG. 31 also
8 shows a moving average 523 of the first trace 521, the average 524 of the first trace, and
9 the target frame size 524 which is the average of the second trace 522. It can be noticed
10 from FIG. 31 that the size of the reduced frames are proportional to the size in bits of the
11 original encoded file. In the case presented in FIG. 31 the rate reduction is done on all
12 the encoded frames including I, B and P type frames. It should be seen that the moving
13 average frame size over about 15 frames for the original-quality MPEG coded video is
14 very smooth and almost constant in time. This property is not required or used in any way
15 in the adaptive bit rate reduction method. For other applications, this property could be
16 exploited for example to ease calculations because the size of the GOP is kept constant
17 by the encoder and combines with the smoothness of the moving average to ensure that
18 the transport stream (TS) bit rate is constant in time (i.e., CBR). Because this property is
19 not required for the adaptive rate reduction method, the method may be performed with
20 the same accuracy for frame sequences that are not encoded for CBR. For example, the
21 method may be used upon I frames removed from original-quality MPEG coded video to
22 generate trick files for which there is no correlation between I frames separated by large
23 time intervals and making prediction of frame size impossible.

FIG. 32 illustrates such a case of independent I frames. FIG. 32 includes a first trace 531 for the I frames from original-quality MPEG coded video, a second trace 532 which is a moving average of the first trace 531, a third trace 533 for the reduced-quality I frames, and a fourth trace 534 which is the target frame size for the reduced-quality I frames. It should be apparent that the moving average frame size in this case is far from being constant and still the reduced rate frame sizes are proportional to the sizes of the original encoded I frames.

FIGS. 33 to 35 show a flowchart of programming of a digital computer for adaptive bit rate reduction of MPEG coded video as introduced above. For example, the programming could be used in the MPEG scaling program (38 in FIG. 1) in the stream server computer (25 in FIG. 1) in the video file server 24 in FIG. 1. In a first step 541 of FIG. 33, the bucket is cleared. In step 542, the first video frame in the original-quality MPEG clip is parsed down to the level of 8x8 DCT blocks. Then in step 543, the DCT coefficient bit rate reduction factor is determined, as will be further described below with reference to FIGS. 36 and 37. In step 544, the first 8x8 block is identified, and a block index (J) is set to zero. In step 545, the first 8x8 block is parsed. If the block has no non-zero AC DCT coefficients, as tested in step 546, then the block cannot be reduced, and execution branches to step 547. If the end of the frame has not been reached, then execution branches from step 547 to step 548, to get the next block and increment the block counter (J), and to parse the next block in step 545. If the end of the frame is reached, then execution continues to step 549. In step 549, if the end of the clip is reached, then the adaptive bit rate reduction for the clip is finished. Otherwise, execution

227 of FIG. 13 and in FIG. 15 could be used, or the approximate largest magnitude scaling as in step 229 of FIG. 13 and in FIGS. 18-19 could be used. If largest magnitude scaling were used, then the first coefficient obtained in step 564 would be the AC coefficient having the largest magnitude. If approximate largest magnitude scaling were used, then the first coefficient obtained in step 564 would be the AC coefficient having the approximate largest magnitude; in other words, it would be the first AC coefficient produced by the hash table sorting procedure of FIG. 19.

In step 565, the procedure determines the number of bits (NBC) for run-length coding for the first non-zero coefficient assuming that the first coefficient will be included in the reduced block. This is a very easy task for the low-pass scaling method in step 225 of FIG. 13, because the number of bits for the run-length coding of an AC coefficient in the reduced block will be the same as the run-length coding of the AC coefficient in the original block. For other scaling methods, the AC coefficient may have to be decoded and re-encoded to determine how many additional bits need to be used for run-length coding to include the AC coefficient in the reduced block. Moreover, for other scaling methods, for a second or subsequent AC coefficient in the block, the addition of a subsequent AC coefficient to the reduced block may change the run-length coding of prior AC coefficients as well as run-length coding for the additional coefficient. After step 565, execution continues to step 571 of FIG. 35.

In step 571 of FIG. 35, the number of bits (NBC) required for encoding and including the non-zero AC DCT coefficient in the reduced block is compared to the number of bit available (NBA) for encoding non-zero AC DCT coefficients in the block. If NBC is not less than or equal to NBA, then execution branches to step 572 to put the

1 remaining unallocated bits into the bucket. Execution loops back from step 572 to step
2 547 of FIG. 33.

3 In step 571 of FIG. 35, if NBC is less than or equal to NBA, then execution
4 continues to step 573. In step 573 the coefficient is transferred to the reduced-rate MPEG
5 coded video. Then in step 574, the number of bits available (NBA) to encode the non-
6 zero AC DCT coefficients of the block is decremented by the number of bits used (NBC)
7 in encoding the non-zero AC DCT coefficients of the block. In step 575, execution
8 branches to step 572 if there are no more non-zero AC DCT coefficients in the block.
9 Otherwise, execution continues from step 575 to step 576 in FIG. 34. In step 576, the
10 next non-zero AC DCT coefficient is obtained in the parsing order, for the
11 implementation of the low-pass scaling method. For other scaling methods, the next
12 largest-magnitude or approximate next largest magnitude non-zero AC DCT coefficient
13 would be obtained in step 576 instead of the next non-zero AC DCT coefficient in the
14 parsing order.

15 FIG. 36 shows a flowchart of a subroutine for determining the bit rate reduction
16 factor (RF) for a reduction from an MPEG source having a known constant bit rate. In a
17 first step 581, the offset rate (S), of bits in the original-quality MPEG source that are not
18 bits of the AC DCT coefficients, is determined. Then in step 582, the coefficient bit rate
19 reduction factor (RF) is computed from the known constant bit rate (BO) and padding
20 (PD) of the original-quality MPEG source, the offset rate (S), and the desired reduced bit
21 rate (BR) of the reduced-quality MPEG coded video. In particular, the coefficient bit rate
22 reduction factor is computed as $RF = (BR - S) / (BO - PD - S)$.

1 FIG. 37 shows a flowchart of a subroutine for determining the coefficient bit rate
 2 reduction factor (RF) for a reduction from an MPEG source having an unknown or
 3 variable bit rate. This subroutine, for example, is called in step 543 of FIG. 33. In a first
 4 step 591 of FIG. 37, the video frame size in bits (VS) is determined, for example from the
 5 parsing of the video frame in the previous step 542 in FIG. 33. Then in step 592 of FIG.
 6 37, a moving average (VAVS) of the video frame size is determined over the last N
 7 frames. For example, $VAVS = (VS(I-N-1) + VS(I-N-2) + VS(I-N-3) + \dots + VS(I)) / N$, where
 8 VS(I) denotes the video frame size of Ith video frame in the original-quality MPEG
 9 source, and N is the number of frames over which the moving average is computed. N
 10 defines a "frame window," which is, for example, the number of video frames over 0.5
 11 seconds worth of video data. In general, the moving average is the result of a low-pass
 12 digital filtering operation upon the video frame size. The low-pass digital filter is
 13 implemented by a subroutine, such as the following:

14 /* Compute Moving Average of up to N samples of VS */
 15 /* J is initially set to N-1 */
 16 /* K is initially set to N-1 */
 17 /* SUM is initially set to zero */
 18 IF (K<0) THEN GO TO CASE2
 19 /* Handle the case at the beginning of the clip where */
 20 /* N samples of VS are not yet available */
 21 SUM ← SUM + VS
 22 VAVS ← SUM/(N-J)
 23 VSBUF(J) ← VS

1 J←J-1
 2 RETURN
 3 /* Handle the case where N samples of VS have been loaded */
 4 /* into the buffer VSBUFF */
 5 CASE2: IF (K<0) THEN K←N-1
 6 SUM←SUM+VS - VSBUFF(K)
 7 VAVS ← SUM/N
 8 VSBUFF(K) ← VS
 9 K←K-1
 10 RETURN

11
 12 In step 593 of FIG. 37, a target average video frame size (VRAVS) is calculated
 13 from an accuracy rate control factor (AR), the desired reduced rate (BR) of the reduced-
 14 quality MPEG coded video, and the video frame rate (FR) according to $VRAVS =$
 15 $AR \cdot BR / FR$. The accuracy rate control factor (AR) is a factor substantially equal to 1 that
 16 can be adjusted on a frame-by-frame basis to control the bit rate of the reduced-quality
 17 MPEG coded video. The accuracy rate control factor (AR) could also be a constant set
 18 sufficiently less than 1 to account for variability in the moving average of the frame rate
 19 so that the bit rate of the reduced-quality MPEG coded video will not exceed the desired
 20 bit rate BR, and stuffing could later be inserted in the reduced-quality MPEG coded video
 21 to obtain precisely the desired bit rate BR.

22 In step 594, the number of bits (BS) in the frame that are not bits of the AC DCT
 23 coefficients is determined, for example, from the parsing of the video frame in the prior

1 step 542 of FIG. 33. In the last step 595 of FIG. 37, the coefficient bit rate reduction
2 factor (RF) is computed as $VRAVS/VAVS$.

3 FIG. 38 shows that a video buffer model could be used for adjusting the bit level
4 in the bucket when the adaptive bit rate reduction procedure 603 (as in FIGS. 33-37)
5 produces a reduced quality MPEG-2 stream 604 in real time from a source 605 of
6 original-quality MPEG-2 coded video. The video buffer model 601 accumulates the
7 video buffer level (VBV) in accordance with the time stamps (DTS, PTS) and program
8 clock reference (PCR) values in the reduced-quality MPEG-2 stream 604 indicating when
9 bits from the stream 604 are added to the video buffer level (VBV) and when bit from the
10 steam 604 are removed from the video buffer level (VBV). In particular, the number of
11 bits in the bucket 602 is incremented when the video buffer level becomes less than a low
12 threshold TH1, and the number of bits in the bucket 602 is decremented when the video
13 buffer level becomes greater than a high threshold TH2. The thresholds TH1 and TH2
14 are selected to prevent underflow of the video buffer level (i.e., VBV becoming less than
15 zero) and to prevent overflow of the video buffer level (i.e., VBV exceeding a maximum
16 size that is greater than TH2). The video buffer model could be implemented as a
17 subroutine that would be called when a frame is encoded and also at the time when the
18 frame would be decoded, relative to the time that the frame is encoded. In this case, for
19 example, the maximum size of the buffer includes a multiplicity of the reduced size
20 frames, the low threshold is larger than the largest reduced frame, and the difference
21 between the size of the buffer and the high threshold is larger than the size of the largest
22 reduced frame. The subroutine could be coded as follows:

23

```

1      /* Video buffer model */
2
3      /* This routine is called when a reduced frame has been encoded. */
4
5      /* Before increasing the video buffer level for the frame just encoded, */
6
7      /* a video buffer queue (VBUFQUEUE) is inspected. The video buffer */
8
9      /* queue has respective entries for frames that have been encoded */
10
11     /* but not yet decoded. Each entry includes a decode time stamp */
12
13     /* indicating when the frame is to be removed from the video buffer */
14
15     /* relative to the loading of the current frame into the video buffer. */
16
17     START: IF (VBUFQUEUE IS EMPTY) THEN GO TO STEP2
18
19     STEP1: GET DECODE TIME STAMP OF ENTRY AT HEAD OF VBUFQUEUE
20
21     IF(DECODE TIME STAMP IS AFTER LOADING TIME OF CURRENT
22     FRAME) THEN GO TO STEP 2
23
24     GET FRAME SIZE (FSHE) OF FRAME OF ENTRY AT HEAD OF
25     VBUFQUEUE
26
27     VBUF←VBUF-FSHE
28
29     REMOVE ENTRY FROM HEAD OF VBUFQUEUE
30
31     GO TO START
32
33     STEP2: IF (VBUF < TH1) THEN BUK←BUK-(VBUF-TH1)
34
35     IF (VBUF < 0) THEN (VBUF ←0; REPORT UNDERFLOW ERROR)
36
37     VBV←VBV+FS
38
39     IF (VBV) > TH2 THEN BUK←BUK-(VBV-TH2)
40
41     IF (VBV>MAX) THEN (VBV ←MAX; REPORT OVERFLOW ERROR)
42
43     PUT FS and relative time stamp of frame on buffer queue

```

1 RETURN

2
3 Although the adaptive bit rate reduction method has been shown in FIG. 38 being
4 used for the real-time production of a reduced-quality MPEG-2 stream 604 from a source
5 605 of original-quality MPEG-2 coded video, the same method can be used in order to
6 reduce the size of an MPEG-2 transport stream file to a given file size either for storage
7 on CD ROMs or for trick file generation.

8 For trick file generation, sometimes there is need to insert freeze frames to
9 compensate for large variations in the frame size. For generating trick files for video-on-
10 demand (VOD) applications, in particular, the file size is generally limited to 10% of the
11 original file size while the target reduced bit rate must be smaller or equal than the bit
12 rate of the original file. When only I frames are used the equivalent bit rate of the I
13 frames of an IBBP encoded clip is 3 to 4 times higher than the bit rate of the encoded file,
14 for example for clips with lots of action encoded at 4 Mbps by a high quality encoder, the
15 equivalent bit rate of the I frames of the clip is 14.5 Mbps. Therefore, the bit rate
16 reduction factor is 3.6, which at such low original bit rate will generate bad quality video
17 frames. To compensate for this large reduction, some of the I frames in the original file
18 are discarded and replaced by P freeze frame that are very small in size. The selection of
19 I frames to discard is done based on the number of bits that will be allocated per block at
20 such a low bit rate as 4 Mbps.

21 In view of the above, there has been described a method of producing reduced bit
22 rate MPEG coded video while achieving uniform proportional video quality reduction
23 proportional to the ratio of the original frame size and the scaled down frame size by

1 proportional reduction of block size and smoothing between frames using a "bucket" of
2 bits. Bits are placed in the bucket when the bits available to be allocated to a block
3 cannot be allocated to the block because there is a rather coarse granularity in the number
4 of bits that can be allocated to a block. A certain fraction of the bits in the bucket are
5 made available for encoding AC DCT coefficients in the block, so that the bits in the
6 bucket after allocation to a block are available for encoding AC DCT coefficients in at
7 least a certain number of following blocks. The method can be applied to a source of
8 original-quality MPEG coded video having an unknown or variable bit rate by computing
9 a moving average of the number of bits per frame over a certain number of consecutive
10 frames. The method improves the quality of the reduced-quality MPEG coded video for
11 a given bit rate by increasing the bit allocation to the video by reducing to a minimum the
12 amount of padding used to precisely achieve the desired bit rate, and by obtaining a
13 substantially uniform degradation in quality over the frames and blocks in the frames.
14 The method can be done in the compressed domain without decoding of the run-length
15 encoded AC DCT coefficients, so that it can be done in real time while streaming MPEG-
16 2 data from a file server at a variable reduced rate to accommodate network congestion.
17 The method can also be used to produce a trick file in parallel with the encoding of an
18 MPEG-2 stream for storage of compressed video in a video file server.

19
20